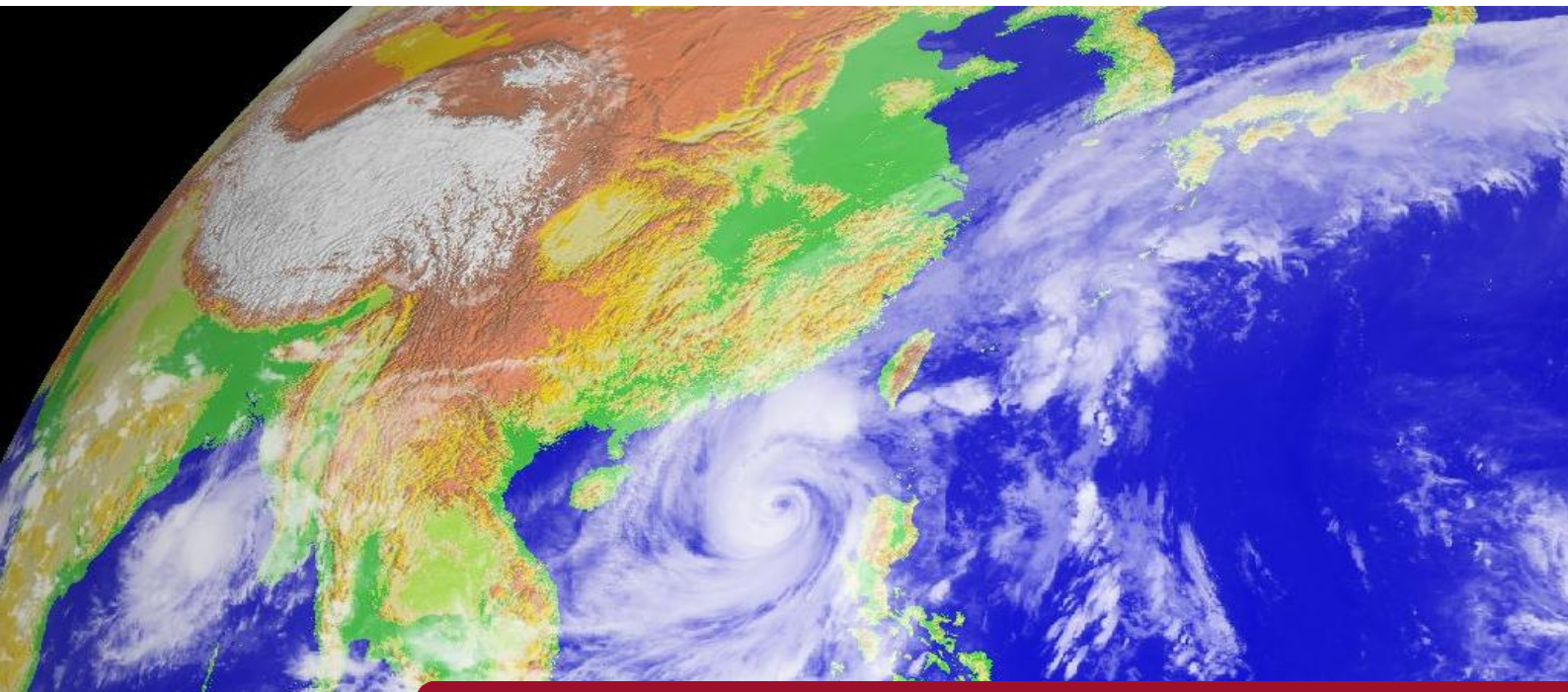
DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Deep Learning for Digital Typhoon

Exploring a typhoon satellite image dataset using deep learning

LUCAS RODÉS-GUIRAO



KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Deep Learning for Digital Typhoon

Exploring a typhoon satellite image dataset using deep learning

LUCAS RODÉS-GUIRAO
lucasrg@kth.se

June 29, 2018

Degree Programme in Electrical Engineering (300 ECTS)
KTH School of Electrical Engineering
Masters Degree in Telecommunications Engineering (120 ECTS)
UPC Barcelona School of Telecommunications Engineering

Principal: Asanobu Kitamoto (National Institute of Informatics, Tokyo, Japan)
Supervisors: Josephine Sullivane (KTH), Xavier Giró-i-Nieto (UPC)
Examiner: Hedvig Kjellström (KTH)
Swedish title: *Deep learning* för Digital Typhoon: Utforska en tyfon satellitbild dataset med *deep learning*

Abstract

Efficient early warning systems can help in the management of natural disaster events, by allowing for adequate evacuations and resources administration. Several different approaches have been used to implement proper early warning systems, such as simulations or statistical models, which rely on the collection of meteorological data. Data-driven techniques have been proven to be effective to build statistical models, being able to generalise to unseen data. Motivated by this, in this work, we explore deep learning techniques applied to the typhoon meteorological satellite image dataset "Digital Typhoon". We focus on intensity measurement and categorisation of different natural phenomena. Firstly, we build a classifier to differentiate natural tropical cyclones and extratropical cyclones and, secondly, we implement a regression model to estimate the centre pressure value of a typhoon. In addition, we also explore cleaning methodologies to ensure that the data used is reliable.

The results obtained show that deep learning techniques can be effective under certain circumstances, providing reliable classification / regression models and feature extractors. More research to draw more conclusions and validate the obtained results is expected in the future.

Resum

Els sistemes d'alerta ràpida poden ajudar en la gestió dels esdeveniments de desastres naturals, permetent una evacuació i administració dels recursos adequada. En aquest sentit s'han utilitzat diferentes tècniques per implementar sistemes d'alerta, com ara simulacions o models estadístics, tots ells basats en la recollida de dades meteorològiques. S'ha demostrat que les tècniques basades en dades són eficaces a l'hora de construir models estadístics, podent generalitzar-se a a noves dades. Motivat per això, en aquest treball, explorem l'ús de tècniques d'aprenentatge profund (o *deep learning*) aplicades a les imatges meteorològiques per satèl·lit de tifons del projecte "Digital Typhoon". Ens centrem en la mesura i la categorització de la intensitat de diferents fenòmens naturals. En primer lloc, construïm un classificador per diferenciar ciclons tropicals naturals i ciclons extratropicals i, en segon lloc, implementem un model de regressió per estimar el valor de pressió central d'un tifó. A més, també explorem metodologies de neteja per garantir que les dades utilitzades siguin fiables.

Els resultats obtinguts mostren que les tècniques d'aprenentatge profundes poden ser efectives en determinades circumstàncies, proporcionant models fiables de classificació/regressió i extractors de característiques. Es preveu que hi hagi més recerques per obtenir més conclusions i validar els resultats obtinguts en el futur.

Sammanfattning

Effektiva varningssystem kan hjälpa till med hanteringen av naturkatastrofer, genom att möjliggöra tillräckliga evakueringar och resursförvaltning. Flera olika tillvägagångssätt har använts för att genomföra lämpliga tidiga varningssystem, såsom simuleringar eller statistiska modeller, som bygger på insamling av meteorologiska data. Datadriven teknik har visat sig vara effektiv för att bygga statistiska modeller och kunna generalisera till ny data. Motiverad av detta utforskar vi deep learning tekniker som tillämpas på tyfonens meteorologiska satellitbild dataset Digital Typhoon". Vi fokuserar på intensitetsmätning och kategorisering av olika naturfenomen. För det första bygger vi en klassificerare för att skilja mellan naturliga tropiska cykloner och extratropiska cykloner och för det andra implementerar vi en regressionsmodell för att uppskatta en tyfons mitttrycksvärde. Dessutom utforskar vi också cleaning methods för att säkerställa att de data som används är tillförlitliga.

Resultaten visar att deep learning kan vara effektiva under vissa omständigheter, vilket ger tillförlitliga klassificerings-/regressionsmodeller och extraktorer. Mer forskning för att dra mer slutsatser och validera de erhållna resultaten förväntas i framtiden."

Acknowledgements

First of all I would like to thank Kitamoto-sensei for the 6 month internship opportunity at the National Institute of Technology. 有難うございました。Back in Stockholm, special thanks to Josephine Sullivane for her significant feedback and to Atsuto Maki, whom I assisted in the Machine Learning course. And in Barcelona, thanks to Xavier Giró-i-Nieto for his insights and help. It was a pleasure and an honor to both work with and learn from each of you.

Secondly, family and friends. Mama i Papa gràcies per la vostra paciència, ajut i confiança. Edu, Marta i Àlex un privilegi créixer al vostre costat. Yaya, caminante, no hay camino, se hace camino al andar. Max, del tiet de l'avió, per una infància llarga. Menció pels untos, els regadeta i els counets. Marc, Albert aquí us torno unes línies. Gràcies per compartir el vostre frikisme amb el meu. Que segueixi. Allan, to our talks and dreams. Shout out to Geeks, Tracy, Nico and Mahin... see you soon guys.

Overall, Japan has been an amazing experience. And this is thanks to the amazing people I have met there. Alex, was a pleasure to share this project and time with you. 14th floor, to those izakayas and combini visits.

I would not be here if it was not for a long list of individuals who influenced me. To those with names and to those anonymous, to those I have known for more than a decade and to those I met recently, to those I will meet soon and to those I will never see again.

Al ĉiuj vi. Dankegon.

Preface

> *When I had thus resouled my selfe, I went a boord of the Shippe of Bengala, at which time it was the yeere of* ***Touffon****, and to vnderstand what this* ***Touffon*** *is: vnderstand, that in the Indies often times, there is not stormes as is in other countries, but euerie ten or twelue yeeres, there is such tempests and stormes, that it is a thing incredible, but to those that haue séene it, neither doe they knowe certaine what yeere it will come.*
>
> **This* ***Touffon*** *is an extraordinary storme at Sea.*
>
> Cesare de Federici, 1588

This quote is from Federici [1] and is the first ever recorded text containing an early version (*touffon*) of the english word typhoon. While the etymology of the word is unclear, Oxford Dictionary of English refers to its Arabic, طوفان (ṭūfān), and traditional Chinese, 颱風 (tai fung, which means "big wind"), counterparts as the possible origins. However, Portuguese or Greek might have also played a roll in the roots of the word Typhoon.

To all victims of natural disasters.

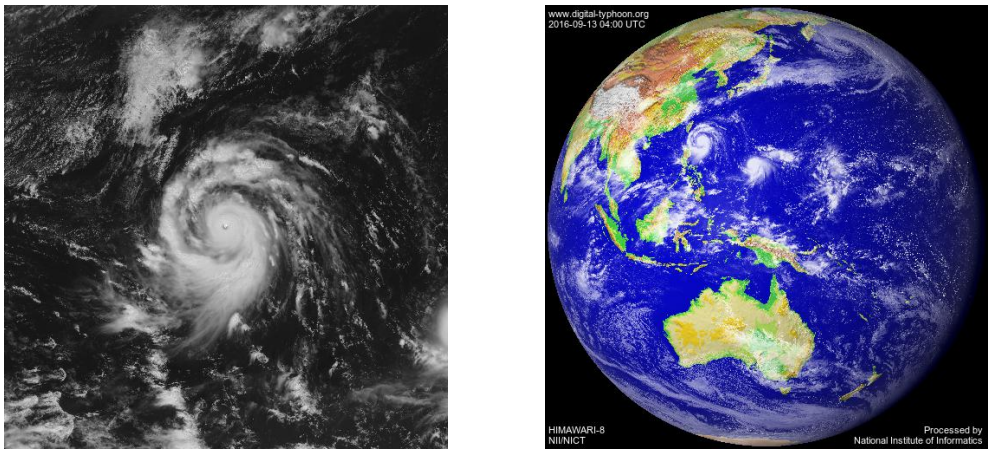
Contents

Chapter 1

Introduction

Tropical cyclones (TC) are rapidly rotating weather systems with, inter alia, low-pressure centre levels and destructive winds, often responsible for heavy rainfalls which can bring floodings. Fastest winds occur in the surroundings of the, apparently tranquil, cyclone eye. Due to their occurrence in tropical and subtropical regions, which are densely populated, they pose a significant threat to lives and infrastructures in these areas. In its most intensified form, a tropical cyclone is referred to as *typhoon* in the northwestern Pacific, *hurricane* in the Atlantic and *cyclone* in the south Pacific or Indian Ocean. Knutson et al. [2] and Emanuel [3] have pointed out that there is evidence indicating a correlation between TC activity and mean global temperatures, presaging that the former might intensify as the latter increase. Therefore, techniques to efficiently forecast and assess the impact of TCs can be extremely helpful towards rapid response in emergency situations. Figure 1.1 illustrates typhoon Meranti (also known as typhoon Ferdie in the Philippines).

Several TC forecasting algorithms have been proposed. These include: Location tracking estimation [4, 5], wind field uncertainty modelling [6], precipitation forecast [7] and TC intensity estimation [8, 9]. In this regard, this thesis focusses on the latter kind of algorithms and continues the work initiated by Chen [10] as part of the *Digital Typhoon* [11] project, which was ideated by Kitamoto-sensei as a means to assist meteorologists in scenarios where TCs endanger society.



(a) Typhoon Meranti region image (b) Full disk image

Figure 1.1: Visible meteorological satellite observation images from 13th September 2016 04:00 (UTC), taken from [11]. Typhoon Meranti has been one of the most intense typhoons ever recorded with peak sustained winds of 120 knots. Meranti caused US$4.8 billion in damage and killed 47 people in Batanes (Philippines), Taiwan and Fujian (China).

1.1 Approach

This work attempts to use state-of-the-art deep learning image-based methods to estimate TC-related parameters given meteorological satellite imagery in the Northwestern Pacific region. In particular, we focus on the difference between extratropical cyclones and tropical cyclones and (ii) on typhoon[1] central pressure and employ CNNs, as their performance in computer vision related tasks has been shown to outperform other traditional approaches [12, 13].

1.2 Challenges

This project tackles a real-world problem, and as such the data has been collected using several sensors and different sources throughout a long period of time. This brings in a highly heterogeneous dataset with some unbalanced and irregular measurements, some of them as

[1]The terms *typhoon* and *tropical cyclone* are used interchangeably throughout this work.

a result of technical failures in the sensors. Since deep learning models fully rely on the quality of the training data, overcoming these issues is key to obtaining satisfactory results. Furthermore, the large amount of data makes that an efficient usage of the computational resources becomes mandatory in order to decrease training times.

1.3 Contribution

So far the use of deep learning in the field of Environmental Informatics has been reduced. However, such techniques are gaining momentum as progress is done in the research community [14, 15, 16, 17]. In this regard, this thesis attempts to join this research inertia and provide new image-based deep learning approaches to improve typhoon parameter estimation.

Our contribution may be summarised as follows:

- Supervision and release of Digital Typhoon dataset.
- Deep learning classification model: Implementation of a model to distinguish between extratropical and tropical cyclones.
- Deep learning regression model: Implementation of a model for typhoon central pressure estimation.
- Development of python library for easy interaction with Digital Typhoon data: `pyphoon` [2]

1.4 Outline

This thesis is organised in six chapters, including the present one which serves as an introduction. Chapter 2 details the origin and scope of the *Digital Typhoon* project and reviews recent related work in the field of Environmental Informatics. Chapter 3 focuses on how the dataset used in this project was built and reveals the different data cleaning

[2] Github repository: http://github.com/lucasrodes/pyphoon, Official documentation: http://lcsrg.me/pyphoon

and pre-processing techniques that have been used. Chapter 4 provides an overview of Deep Learning theory and its most relevant features to this work. Chapters 5 presents the experiments done for this work along with their corresponding results. Finally, chapter 6 concludes this report with a brief discussion of the obtained results and a batch of eventual future research paths.

Chapter 2

Background

In this chapter, we first explain the details about the *Digital Typhoon* project and later present some work done in the field of Environmental Informatics.

2.1 Digital Typhoon

The Digital Typhoon project was conceived as a platform where typhoon-related information could be easily delivered and processed in nearly real-time, serving as a spatiotemporal scientific database. It is an example of an application of *meteoinformatics* to large-scale real-world issues. This project contains large sets of typhoon satellite imagery from nearly 40 years, enabling researchers to implement and test various informatics-based approaches, such as pattern recognition, computer vision or data mining. Some of these methods have been explored by researchers in an attempt to better understand the nature of typhoons and potentially provide tools for socially relevant problems (e.g. early warnings). Furthermore, in scientific communities, such as earth sciences, large amounts of data are continuously being generated, making it imperative an efficient and scalable implementation of algorithms in order to rapidly process large volumes of data. While it was initially meant to be a database for Northwestern Pacific tropical cyclones, nowadays the Digital Typhoon also provides Southwestern Pacific tropical cyclone imagery. However, this work only considers the northern hemisphere data.

2.1.1 History

According to the official website[3], the Digital Typhoon project was initiated the 22nd of April in 1999. One of the motivations was the fact that a typhoon image collection could be constructed by performing a simple pre-processing step such as cutting the image according to the *typhoon eye*. Later, in a meeting on the 13th of May 1999[4] the foundation for the future development of the project was laid. First components and goals of the project were mentioned, such as automatic image segmentation, typhoon image indexing and search (to name a few). However, these ideas first became public in [18], where the author introduced a typhoon image database with a content-based search. In addition, this work provided an analysis of typhoon cloud patterns and presented an algorithm to automatically classify clouds in seven categories. Later, some works presented some enhancements on the initial database structure [19, 20, 21]. The project rapidly leant towards exploring data mining techniques (e.g. statistical methods) to analyse the images of Digital Typhoon database [22, 23, 24]. Nowadays, there is an ongoing development on this project as satellite image sources change and new algorithms in the informatics community emerge. In particular, deep learning has recently raised some interest [10]. The project also provides live notifications via its Twitter account [5].

Definition: ***Typhoon eye***

Rougly circular with an average 30–65 km diameter, the eye is located at the centre of strong tropical cyclones. It has mostly calmed weather, being the blue sky even visible from therein. It is encircled by the *eye wall*, a ring that separates the central region from the surrounding area, which has the highest surface winds in the tropical cyclone. The cyclone's lowest barometric pressure occurs in the eye and can be as much as 15 percent lower than the pressure outside the storm. There are several hypothesis on how the eye is formed [25, 26].

[3] http://agora.ex.nii.ac.jp/digital-typhoon/history/ten-years/index.html.ja

[4] See the details of 13th May 1999 meeting at http://agora.ex.nii.ac.jp/digital-typhoon/history/ten-years/19990513.html.ja (Japanese)

[5] Digital Typhoon Twitter account: https://twitter.com/digitaltyphoone?lang=en (English version)

2.2 Related Work

TC intensity estimation is usually based on *maximum sustained wind*, which is a typhoon-related parameter that is usually measured using satellite imagery and, when available, radar imagery or land/sea/air reconnaissance observations. However, centre pressure values might be also used to estimate TC intensity.

Definition: ***Maximum sustained wind***

It serves as an indicator of the intensity of a tropical cyclone. Maximum sustained wind is defined as the highest average wind speed anywhere in the tropical cyclone (typically within the eyewall) over a certain time span (in this work we use ten-minute span).

When it comes to using satellite imagery, Dvorak Technique [27, 28, 29] has been widely used to estimate typhoon intensities. This technique assumes that there is a correlation between the cloud patterns and the TC intensity. Figure 2.1 illustrates the different cloud patterns contemplated by Dvorak technique. Furthermore, it also provides some guidelines on how to interpret day-by-day changes in the cloud patterns. While Dvorak Technique has been the *de facto* image-based technique for a long time it has some drawbacks. In particular, it is based on subjective pattern distinction, therefore limiting the expressiveness of the model. In this regard, some improvements have been proposed to overcome this subjectivity by reducing human intervention [30, 31, 32, 33]. Nevertheless, these still depended, to some degree, on the user expertise.

Deviation angle variance technique (DAVT) is yet another family of techniques used to determine the intensity of a TC by means of a statistical analysis of the gradient of the brightness in infrared satellite imagery. In [34] and [35] the authors use DAVT for the North Atlantic and North Pacific oceans, respectively. This technique requires some parameters to be adjusted according to the geographical region being explored and hence might not generalise well to new locations.

Non-image-based methods have also been extensively explored for TC intensity forecast. Chu [36] designed a regression model to forecast TC intensity for up to 72 hours. In [8], the authors proposed a battery of 5-day tropical cyclone intensity forecast models based on climatology and persistence for the Atlantic and North Pacific. Emanuel et al.

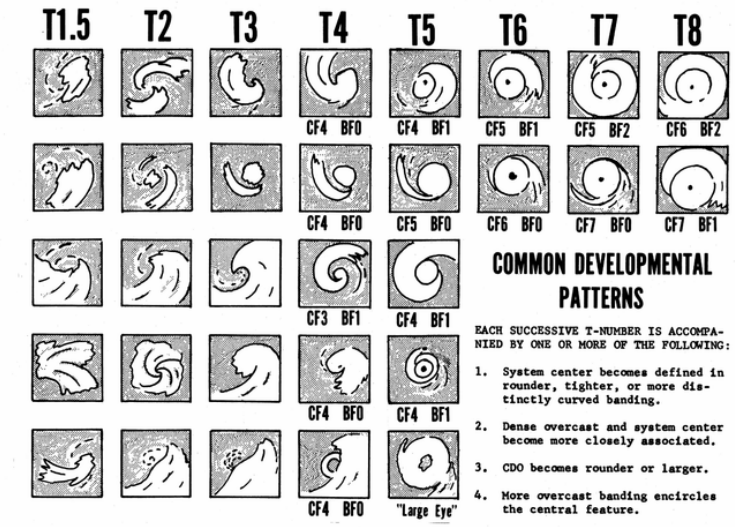


Figure 2.1: Cloud patterns and intensities according to Dvorak technique [27].

[37] introduced a simple coupled ocean-atmosphere model, which outperform other numerical models and remained competitive with statistical methods. Multiple linear regression models have been proven to produce reliable intensity estimation forecasts through 4 days, based on a 48-storm verification [9]. These methods rely on past data to predict future data and thus should be considered for further implementations of a time-series-based TC intensity prediction model.

In the search for an objective image-based region-independent method, we observe an increasing research trend towards data-driven approaches in the computer science community. Chen [10] designed and implemented a deep learning based model, illustrated in figure 2.2, which estimates, out of 5 possible classes, the *intesity class* of a typhoon given its satellite image from the Digital Typhoon dataset. They used a VGG-16 [38] pre-trained model and applied it to the dataset, achieving a 39.72% test accuracy. Moreover, they were able to increase the test accuracy up to 63.92% by combining the pre-trained model with a long short-term memory (LSTM) network. This highlighted the importance of time information in the analysis and estimation of typhoon intensity, which is reasonable since Typhoons are natural phenomena that evolve over time. In [17] the authors introduced a purely CNN-based model to estimate the intensity of tropical cyclones (i.e. classify

them into classes) in Atlantic and Pacific regions. Using 98 cyclone sequences, they achieved a top-1 accuracy of 80.66% on hurdat2 dataset[6]. Recently, Miller, Maskey, and Berendes [16] claim to have achieved a performance of 86.4% top-1 accuracy and an RMSE of 10.0 kt. It is worth mentioning that both [17] and [16] owe their deep network architecture to the work of Krizhevsky, Sutskever, and Hinton [12].

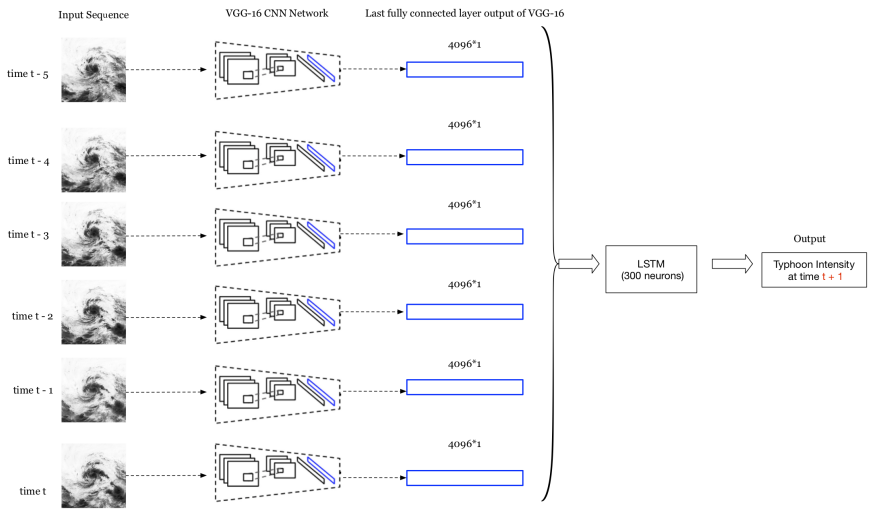


Figure 2.2: Network architecture from [10]. The main idea resided in using a sequence of 6 typhoon satellite images to (i) estimate the typhoon intensity at time t and (ii) predict the typhoon intensity at time $t+1$. They combined a pre-trained model (VGG-16) to extract relevant features from the input images with an LSTM network to use the typhoon temporal evolution. Using transfer learning with a pre-trained model can save hours of computation by making use of pre-learned features. However, if the new dataset substantially differs from the one used to pre-train the model, the performance of the overall model might be affected. In our particular case, the Digital Typhoon data contains a very concrete type of data and thus makes transfer learning from standard pre-trained models not very suitable.

[6] http://www.nhc.noaa.gov

Chapter 3

Dataset

In this chapter, we first present the two main types of data that constitute the Digital Typhoon dataset. Next, we propose some practical adjustments to the dataset to efficiently use the available computational resources.

3.1 Dataset components

The dataset consists of two main components: (i) Meteorological satellite images and (ii) best track data. The former provides the typhoon images while the latter contains all the metadata parameters, such as wind speed or centre pressure levels.

3.1.1 Meteorological Satellite Images

Different sources have been used throughout the existence of the Digital Typhoon project. In particular, between 1978 and 1995 satellite images (VISSR) from the Japan Meteorological Business Support Center (JMBSC) were employed. Next, from 1995 until 2003, satellite images received at the Institute of Industrial Science (IIS), University of Tokyo were used. Finally, since 2003 the Japan Meteorological Business Support Center (JMBSC) and MTSAT-1R (Himawari 6) have been the main source of imagery.

The images are obtained from geostationary satellites, allowing a coverage of Northwestern and Southwestern Pacific regions. Table 3.1 shows the number of *typhoon sequences* within Digital Typhoon dataset

depending on the basin (i.e. hemisphere). Figure 3.1 illustrates typhoon sequence 199303.

Definition: ***Typhoon sequence***

A typhoon sequence is an array of image frames that illustrate the state progression of a certain typhoon as time passes. The observation frequency is, typically, of one hour, however, some sequences present temporal gaps which cause the observation frequency to vary. The images within a sequence can have multiple formats: visible, infrared etc. There can be multiple sequences referring to similar time periods, as several typhoons may occur simultaneously.

Table 3.1: Number of sequences and images depending on the region. The first sequence was captured on the December 1st of the year 1978.

Basin	Sequences	Images
Northwestern Pacific	972	164,631
Southwestern Pacific	397	53,565
Total	1,369	218,196

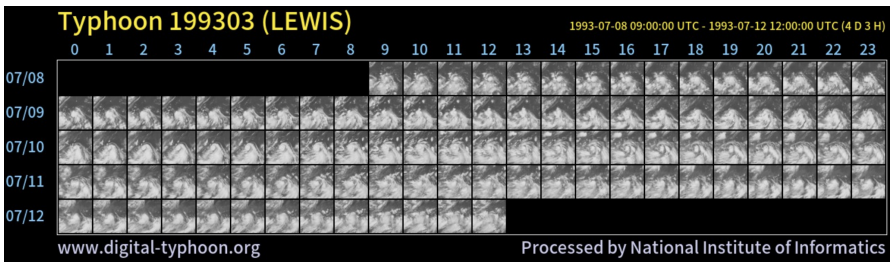


Figure 3.1: Grid with the image frames from typhoon sequence 199303, taken in July 1993 (UTC time). Image from [11].

Definition: ***UTC***

UTC is short for Coordinated Universal Time, a world 24-hour time standard used to regulate time. UTC is determined using two main sources: International Atomic Time (TAI) and Universal Time (UT1). It can be often interchangeable with Greenwich Mean Time (GMT), but GMT is no longer considered a standard but rather a time zone, used in few African and Western European countries.

Image Format

Satellites provide visual-band and infrared-band images, which, depending on the year, present slight differences in their frequency bands. For our work, we have used images from Infrared sensors, which have dimensionality 512×512 and have been pre-processed so that the typhoon eye is located in the centre of the image, as figure 1.1a illustrates. Furthermore, they have been calibrated in order to take into consideration non-linearities such as the effects of the atmosphere and other interference. As a result, the pixel intensity directly measures the temperature in Kelvin (K) of either cloud, land or sea surface.

3.1.2 Best Track Dataset

Best track dataset provides specific parameters of a typhoon every 6 hours. These data come from national meteorological agencies and, as such, is backed by meteorology experts, which makes it comprehensible to regard this dataset as nearly ground truth. Data for the Northwestern Pacific was initially obtained from Japan Meteorological Agency, then the Ohio State University, and currently the Unisys Weather Hurricane. Regarding the Southwestern Pacific, the data is distributed from Bureau of Meteorology, Australia.

Best track data provides several parameters for a given typhoon at a given time. In the following, we briefly summarise those parameters which we found were the most relevant to our work. However, a full list can be found in the JMA website [7].

Timestamp

Timestamp consists of 4 elements: year, month, day and hour.

Latitude, Longitude

Information on the geolocation of the typhoon is provided by its latitude and longitude in degrees. Figure 3.2 illustrates the recorded typhoon activity in the Northwestern Pacific region.

[7] Format of RSMC Best Track Data: http://www.jma.go.jp/jma/jma-eng/jma-center/rsmc-hp-pub-eg/Besttracks/e_format_bst.html

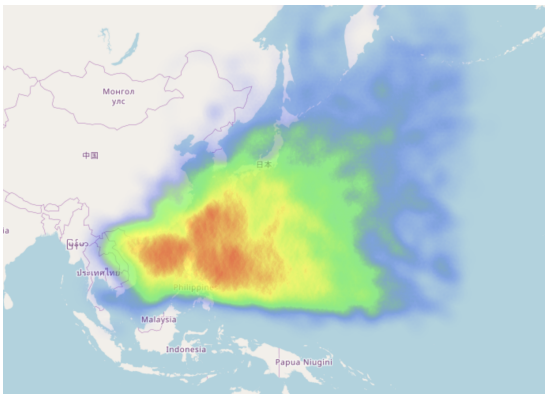


Figure 3.2: Heatmap on the typhoon activity in Northwestern Pacific since 1978. It highlights that areas such as Phillipines have been largely affected by TC activity.

Maximum sustained wind speed

Specifies the maximum 10-minute sustained wind from a typhoon. It is measured in *knots* and oscillates from 0 to 140 knots. However, this data is not available for all samples, particularly for class 2 and 6 samples (to be explained in the following), which forced us to discard it for parameter estimation.

Definition: ***Knot (kt)***

A knot is a non-SI unit used to measure speed in meteorological fields, air and sea navigation. It is equivalent to one nautical mile per hour.

Central Pressure

It measures the pressure at the typhoon eye, which presents the lowest atmospheric pressure in the typhoon. In the Digital Typhoon dataset, the pressure ranges from 870 to 1018 hPa. Figure 3.3 shows the histogram of the pressure values. The Digital Typhoon website provides various charts to visualise best track data, including pressure evolution, as figure 3.4 shows.

Class

Digital Typhoon dataset contains data related to two very distinguished natural phenomena: typhoons (a.k.a. tropical cyclone) and *extratropi-*

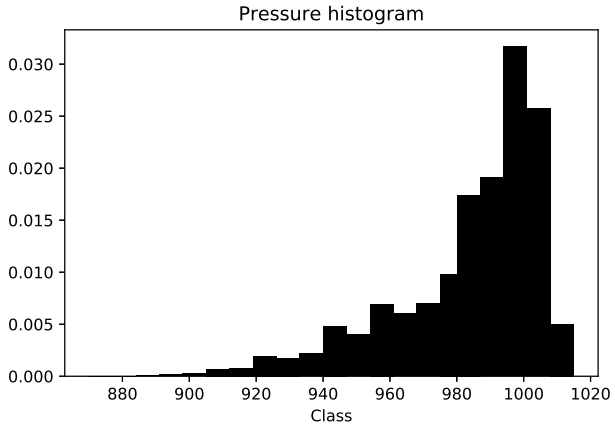


Figure 3.3: Centre pressure histogram of the dataset samples. Resolution of 7 hPa.

cal cyclones. The former can be further classified, based on their wind speed and central pressure, into subcategories which measure their intensity and damage capacity. All categories are listed in table 3.2.

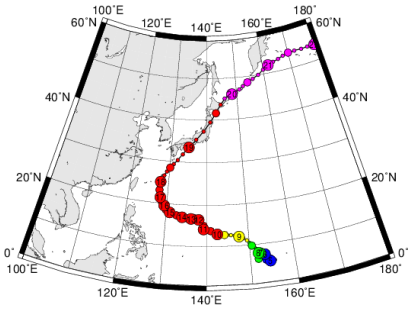
Table 3.2: Categories of Digital Typhoon data based on maximum 10-min sustained wind measured in *knots*.

Category	**Maximum Sustained Wind (10-min)**
Tropical Depression (2)	≤ 33 kt
Tropical Storm (3)	$34 - 47$ kt
Severe Tropical Storm (4)	$48 - 63$ kt
Typhoon (5)	≥ 64 kt
Extratropical Cyclone (6)	-

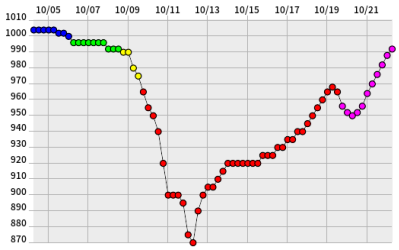
Definition: ***Extratropical Cyclone***

The main difference between an extratropical cyclone and a tropical cyclone resides in their temperature at the eye. This temperature is colder in the case of extratropical cyclones and warmer in the case of tropical cyclones compared to neighbouring regions at the same altitude. Due to these differences, tropical cyclones have their heaviest winds close to Earth's surface, whereas extratropical cyclones tend to have their strongest winds near the tropopause (i.e. 12 km altitude).

Figure 3.5 illustrates the class distribution of the dataset. In the



(a) Latitude and longitude.



(b) Central pressure (UTC time).

Figure 3.4: Data charts from typhoon sequence 197920, also known as Typhoon Tip (or Typhoon Warling in the Philippines) during October 1979. Tip was the largest and most intense tropical cyclone ever recorded, with the worldwide record for lowest central pressure with 870 hPa and maximum winds 140 kt. Images from [11].

ideal training scenario, it should preferably be balanced, such that the model can equally learn from all different classes. However, in real-world datasets, like ours, balanced datasets are hardly observed and thus some techniques are used to overcome this complication.

3.2 Re-designing the dataset

When it comes to practicalities, the format of the data and the way it is stored plays an essential role. This has a substantial impact on the training time. Furthermore, the reliability of the dataset is essential as the deep learning algorithms fully lean on data. Hence, exploring the data to detect possible corrupted fields becomes a must. This process has lead to the implementation of the python library `pyphoon`, which is available under GNU General Public License v3.0[8].

3.2.1 Data cleaning

Observation frequency alignment

As the reader might have already identified, the observation frequency in the image and the best track datasets differ. In particular, the former

[8]Github repository: http://github.com/lucasrodes/pyphoon, Official documentation: http://lcsrg.me/pyphoon

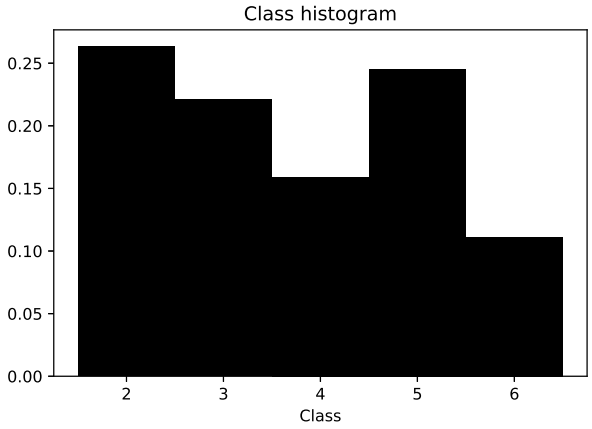


Figure 3.5: While class 2 samples are responsible for more than 25% of the whole dataset class, class 4 and class 6 only account for nearly a 15% and 10%.

uses a 1-hour while the latter uses a 6-hour observation rate. To this end, the best track data has been interpolated so as to obtain the set of parameters every hour. Cycloid interpolation was used for approximating the eye movement of the typhoon and cubic spline interpolation for the rest of the parameters [18].

Detecting corrupted images

With more than 150,000 images in the dataset, it is conceivable that some of the images contain some errors. This is mainly due to some technical failures in the recordings. To find corrupted images we defined a tolerance range of pixel values (i.e. temperatures), and regarded images containing pixel values outside of this range as corrupted. We used the range [160, 310] K (Kelvin), which was set after carefully exploring the histogram of pixel values of all images (see figure 3.6) in the dataset and by using meteorological-domain knowledge.

Mathematically, we define the *error function* $e(\cdot)$, which outputs 1 if a pixel is corrupted (in our case out of range) or 0 otherwise. Although this heuristic is simple and might be regarded as rather naive, it is able to successfully detect numerous corrupted images. The first row in figure 3.7 illustrates some examples of corrupted images detected by our algorithm.

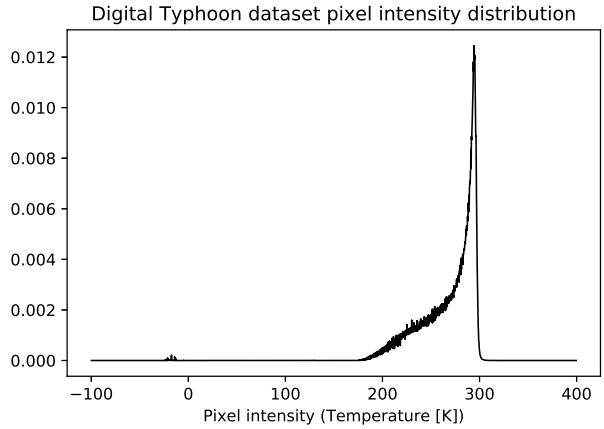


Figure 3.6: Histogram of the pixel values (i.e. temperature [K] values) from the IR satellite images in Digital Typhoon dataset. We note that the distribution is not symmetric with the mean at 269.15 K and a standard deviation of 24.14 $\sqrt{\text{K}}$. The peak around 294 K might be explained by the sea surface temperature since most satellite images contain several pixels from the ocean and land. The long-term global average surface temperature of the oceans is 16.9^oC [39], but in tropical areas, this might increase up to the range of $[22, 27]^o$C, which is roughly equivalent to 294 K.

Correcting corrupted image files

Once potential corrupted images have been detected, there are several possible approaches to estimate their corrected counterparts. Our particular approach consisted in interpolating image frames from the same typhoon close in time. In the following, we will use mathematical notation to rigorously describe the process of correcting an image.

Let $S = [I_1, I_2, \ldots, I_K]$ be a typhoon image sequence, where I_k is the k:th image frame in the sequence S and K is the number of image frames within the sequence. We have that $I_k(n, m)$ denotes the (n, m) pixel value for $0 < n, m < N$, where N is the width and height of the image (we deal with square images). Using the error function e, we define the corrupted region E_k from I_k as the set of pixel locations with corrupted pixel values,

$$E_k = \big\{(n, m)\big\} \Big| e\big(I_k(n, m)\big) = 1. \tag{3.1}$$

If $E_k \neq \emptyset$, which means that there are some corrupted pixels, we look in preceding and following image frames for non-corrupted pixel values located within this same region E_k and use them to generate new values for I_k. To better understand this procedure, let us focus on just one pixel, which makes sense as the correction is performed pixel-wise. Let us suppose that pixel $I_k(n_0, m_0)$ is corrupted. The fixed value is computed as

$$\hat{I}_k(n_0, m_0) = \frac{e^{-d(I_k, I_{k-\delta_1})} I_{k-\delta_1}(n_0, m_0) + e^{-d(I_k, I_{k+\delta_2})} I_{k+\delta_2}(n_0, m_0)}{e^{-d(I_k, I_{k-\delta_1})} + e^{-d(I_k, I_{k+\delta_2})}}, \quad (3.2)$$

where δ_1 and δ_2 denote the number of frames checked backwards and forwards, respectively, until a non-corrupted pixel value is found and $d(\cdot, \cdot)$ provides the distance (in time) between two frames. In short, we find two frames with non-corrupted pixel value at position (n_0, m_0), weight them depending on how close in time they are to I_k and normalise the resulting value according to the interpolation weights. This method could be understood as an optimisation problem, where δ_1 and δ_2 are to be minimised such that $I_{k-\delta_1}(n_0, m_0)$ and $I_{k+\delta_2}(n_0, m_0)$ are non-corrupted values.

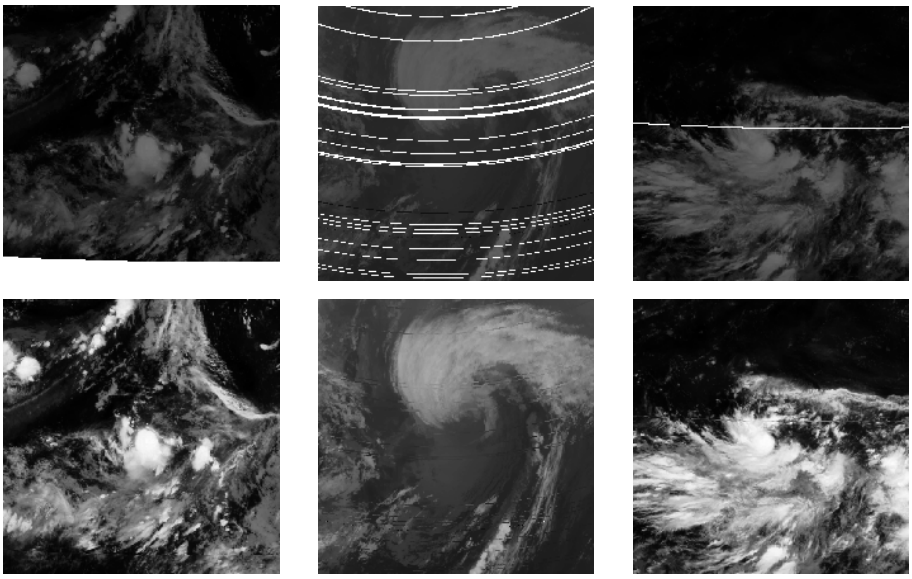
Second row in figure 3.7 illustrates some examples of corrected corrupted images. We note that sequence 198404 was discarded due to serious corruption.

Image reshape

Decreasing the dimensionality of the images reduces the number of pixels (i.e. features) of the input samples and thus less summation-multiplication operations are required. This significantly relaxes the computational power demands. In this work we have scaled down the images by a factor of 2, yielding to images of shape 256×256. It is worth noting that most widely used models, such as [12], use similar dimensions.

Incomplete typhoon sequences

Some typhoon sequences present time intervals greater than one hour without an image from the typhoon. This leads to an efficient observation frequency higher than one hour. Moreover, not having a constant time separation between frames hinders a smooth application of



(a) 26th of June 1987 at 14:00 PST (from Typhoon Sequence 198704).

(b) 6th of September 1982 at 21:00 PST (from Typhoon Sequence 198215).

(c) 20th of June 1982 at 03:00 PST (from Typhoon Sequence 198205).

Figure 3.7: Some examples of corrupted images from Digital Typhoon Dataset. The first row illustrates corrupted images and second row their corrected counterparts. Corrupted image (a) contains a pixel region in the lower part of the frame with very low values. Similarly, corrupted images (b) and (c) show some lines with low values. By correcting these errors we can draw on the corrected images, which increases the number of available samples for training.

time-series models. To this end, two possible solutions have been considered.

1. **Image interpolation**: This consists in using two typhoon images to estimate typhoon images in between. This approach is simple and may be effective for short gaps (i.e. 1 hour), where just one or two images have to be estimated. However, it presents some dissonant estimates for longer gaps, where several images are to be generated.

2. **Divide sequences in sub-sequences**: Here, no synthetic images are generated, leaving the original data untouched. Instead, the

typhoon sequence is divided into sub-sequences without temporal gaps.

After some consideration, we combined both options. In particular, option 1 was used in cases where there was up to 2-hour gap. For the rest of the cases, the typhoon sequence was split into shorter subsequences. Sub-sequences shorter than 4 images were discarded. This lead to a new dataset, which was only used in tasks that required temporal information.

3.2.2 Data integration

Initially, the data acquired for this work was handed in two separated file formats. On the one hand, the satellite image dataset was given as a set of 971 folders, one per typhoon sequence, with their corresponding images within as HDF5 files. On the other hand, the best track data was given as a TSV file per typhoon sequence.

To ease data analysis, we used the aforementioned library `pyphoon`. In addition, we planned on creating a database to monitor different versions of images (original version, corrected version etc.) and further analyse best data. This approach would allow for easy updates and maintainance of the dataset. In this regard, methods to easily export data from the database would be required.

We note that this structure is still under development and might suffer some modifications in the near future.

Chapter 4

Deep Learning. An Overview

Contrary to task-specific algorithms, machine learning algorithms attempt to learn data representations which can be generalised to other domains. There are three main approaches:

- Supervised learning: Consists of finding a model that recreates a mapping $f : x \rightarrow y$, where x is the input and y denotes the target value.
- Unsupervised learning: The model tries to learn from unlabeled data. Clustering is one of the most well-known unsupervised techniques.
- Reinforcement learning: A reward signal gives feedback of success to an agent.

In this work, we mainly focus on supervised approaches.

Deep learning can be regarded as a subset of machine learning, which essentially consists of *artificial neural networks* (ANNs) with a large number of parameters. Such methods have been gaining a lot of momentum since its Hinton, Osindero, and Teh [40] presented an efficient manner of training such deep networks. But the biggest breakthrough happened in 2012 when Krizhevsky, Sutskever, and Hinton [12] reduced the error rate in the Large Scale Visual Recognition Challenge(LSVRC) from a 26% to an astonishing 16% using a deep neural network. This work laid the foundations for modern deep learning and, although some improvements have been made since then, most

of the architecture components are still in use nowadays.

In the following, we review some of the key ideas and main components of ANNs in the context of deep learning. We cover different aspects which are later viewed in this work. In particular, we first review different ANN architectures and later briefly present how these networks are optimised to perform a specific task. This is, by no means, an exhaustive coverage of Deep Learning or ANNs and, hence, for a broader view, the reader should refer to the available literature [41, 42, 43, 44].

4.1 Artificial Neural Networks

ANNs are inspired by the behaviour and functioning of the biological neurons in the animal brain. These attempt to model the interactions in brains by a set of nodes (neurons) mutually connected through simplified synapses. A given artificial neuron processes an input signal and propagates it to its neighbouring artificial neurons, which yields to a specific network state. There are multiple variants of ANNs, depending on which is the topology of the connections between neurons.

The core logic of ANNs is represented by their fundamental component, the *perceptron*, which is a simplified representation of how biological neurons operate. In the following, we briefly describe this unit and later present different - and widely used nowadays - ANN topologies.

4.1.1 Perceptron

The perceptron was introduced by Rosenblatt [45], where they reflected on the understanding the capability of perceptual recognition, recall and thinking by higher organisms. In their work, authors devised a simple mathematical representation of the perceptron, which has thenceforth has sat down the foundations of ANNs. Figure 4.1 illustrates a block diagram of a perceptron.

We can summarise the perceptron operation on an input vector $\boldsymbol{x}$

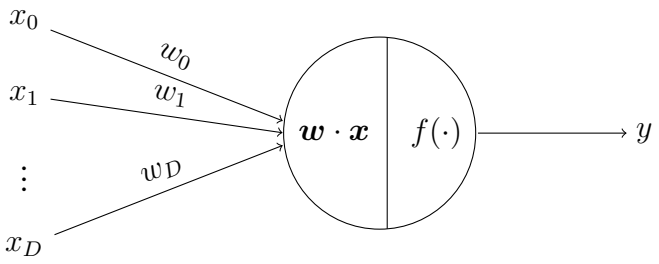


Figure 4.1: In a perceptron, a D dimensional input sample $\boldsymbol{x} = [x_0, x_1, \ldots, x_{D-1}]$ is linearly combined with a weight vector of same size $\boldsymbol{w} = [w_0, w_1, \ldots, w_{D-1}]$. The weight vector $\boldsymbol{w}$ is representative of the perceptron itself and its values are "chosen" to perform some specific task on the input data. After the linear combination, the scalar value $\boldsymbol{w} \cdot \boldsymbol{x} = \sum_{d=0}^{D-1} w_d x_d$ is passed through a non-linear function $f(\cdot)$, which outputs either 0 or 1 based on its sign. We note that, for notation purposes, $x_0 := 1$ is used to include the biases as weights.

as

$$f(\boldsymbol{w} \cdot \boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} > 0 \\ 1 & \text{otherwise} \end{cases}, \tag{4.1}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ is the weight vector, w_0 is the bias term and $\cdot$ denotes the scalar product.

In supervised tasks, which is our main focus here, an error measure (commonly referred to as loss function L) is typically defined. To this end, the weights of the perceptron are such that the error measure is minimised (and thus the performance is maximised), such that the operation in (4.1) performs a desired task on the input data. How the error measure is minimised is left for later sections.

4.1.2 Feed-Forward Neural Networks

Feedforward networks are the simplest type of ANNs and were the first ones that were introduced. Essentially these networks are formed by concatenating densely connected layers of perceptrons. Strictly speaking, no perceptrons are used and we instead talk about *units* or *neurons* since the non-linear activation function may be more complex than the one presented in (4.1). In such networks the information only flows one way (forward), so that neurons in layer $k + 1$ only depend

on the activity on previous layers. In figure 4.2, a D dimensional input sample $\boldsymbol{x} = [x_0, x_1, \ldots, x_{D-1}]$ is first multiplied with a weight matrix $\boldsymbol{W}$ and later undergoes a non-linear function f, i.e. $\boldsymbol{h} = \boldsymbol{f}(\boldsymbol{W} \cdot \boldsymbol{x})$. Note that here $\boldsymbol{f}$ denotes element-wise application of non-linear function f, i.e.

$$\boldsymbol{h} = \boldsymbol{f}(\boldsymbol{W} \cdot \boldsymbol{x}) = [f(\boldsymbol{W}_{1,:} \cdot \boldsymbol{x}), f(\boldsymbol{W}_{2,:} \cdot \boldsymbol{x}), \ldots, f(\boldsymbol{W}_{K,:} \cdot \boldsymbol{x})], \tag{4.2}$$

where $\boldsymbol{W}_{k,:}$ is the k:th row of matrix $\boldsymbol{W}$. Therefore, this operation can be seen as a set of K linear combinations followed by non-linear function. The obtained new representation $\boldsymbol{h}$ is then passed through the same process leading to the next layer's input representation, which in the example from (4.2) is the network output, i.e.

$$\hat{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{V} \cdot \boldsymbol{h}) = [g(\boldsymbol{V}_{1,:} \cdot \boldsymbol{h}), g(\boldsymbol{V}_{2,:} \cdot \boldsymbol{h}), \ldots, g(\boldsymbol{V}_{M,:} \cdot \boldsymbol{h})], \tag{4.3}$$

where $\boldsymbol{V}_{m,:}$ is the m:th row of matrix $\boldsymbol{V}$ and M is the number of units in the layer. Combining (4.2) and (4.3) the overall network operation can be summarised as

$$\hat{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{V} \cdot \boldsymbol{f}(\boldsymbol{W} \cdot \boldsymbol{x})). \tag{4.4}$$

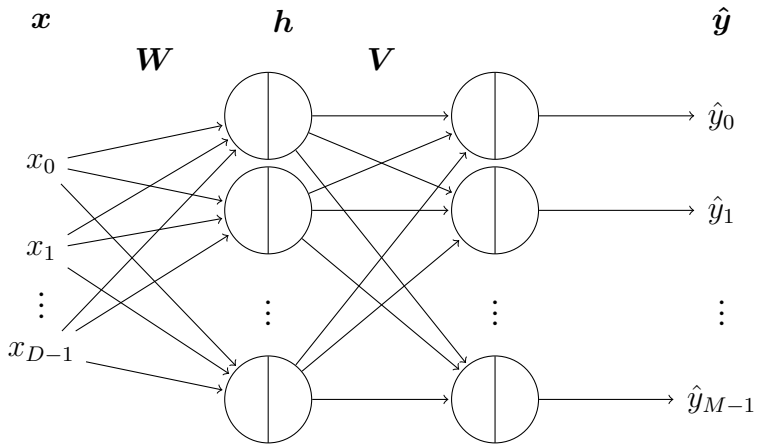


Figure 4.2: Single hidden layer feed forward neural network. From an D-dimensional input vector $\boldsymbol{x}$ we obtain a new M-dimensional vector $\hat{\boldsymbol{y}}$, which aims to resemble the ground truth vector $\boldsymbol{y}$.

Such an architecture from figure 4.2 allows for supervised tasks such as classification, where the output represents a category index

(typically using one-hot-encoding [46]), or regression, where the network maps an input value to a real value. In classification tasks, the output layer is normalised using the softmax activation, which is described in the following.

Activation functions

In the original perceptron, a thresholded non-linear function was used, as presented in (4.1). This, however, has a big drawback: It is not derivable. As we will later see, to update the weights from perceptron units we need to backpropagate information about the performance (somehow measured by means of a loss function) and, in mathematical terms, this means, precisely, derivating. To this end, alternative activation functions have been proposed, which we show some of them in figure 4.3.

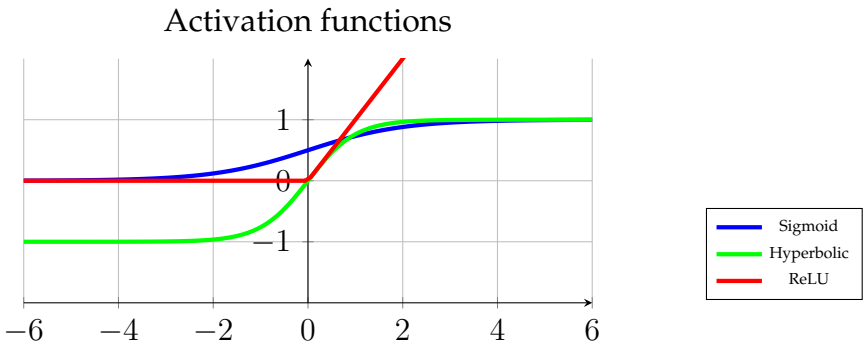


Figure 4.3: A graph with the different activation functions commonly used in machine learning tasks. Sigmoid and hyperbolic tangent saturate as the input moves apart from the origin. On the contrary, ReLU just sets negative inputs to zero.

One of the most commonly used activation functions is the sigmoid function, which squashes a value within the range $(0, 1)$ and is defined as

$$f(\boldsymbol{w} \cdot \boldsymbol{x}) = \frac{1}{1 + e^{\boldsymbol{w} \cdot \boldsymbol{x}}}. \tag{4.5}$$

However, sigmoid outputs are not zero centred, which can lead to undesirable *zig-zagging* while optimising the weights. Thus, the hy-

perbolic tangent function (in (4.6)) was proposed, which squashes the outputs within the range $(-1, 1)$ instead.

$$f(\boldsymbol{w} \cdot \boldsymbol{x}) = \tanh(\boldsymbol{w} \cdot \boldsymbol{x}) = \frac{1 - e^{-2\boldsymbol{w}\cdot\boldsymbol{x}}}{1 + e^{-2\boldsymbol{w}\cdot\boldsymbol{x}}} \tag{4.6}$$

Nevertheless, both hyperbolic tangent and sigmoid have saturation zones which, particularly in very deep networks, yields to the problem of *vanishing gradient* [47] and thus hinders the learning process. In this sense, the rectified linear unit (ReLU) activation was introduced. This activation performs a very simple operation, i.e.

$$f(\boldsymbol{w} \cdot \boldsymbol{x}) = \max(0, x), \tag{4.7}$$

and accelerates the convergence of the optimisation process. This activation function is the usual choice in deep networks, also in this work.

In classification tasks, after the last dense layer a so-called softmax layer is used. This layer normalises the output vector such that its components add up to one. This way, the output values can be understood as the likelihood probability of the input belonging to a certain class [48], i.e.

$$p(\text{class}(\boldsymbol{x}) = m_0 | \boldsymbol{x}, \boldsymbol{W}, \boldsymbol{V}) \propto \frac{e^{-\hat{y}_{m_0}}}{\sum_{m=0}^{M-1} e^{-\hat{y}_m}}, \tag{4.8}$$

where $\hat{y}_{m_0}$ is the m_0:th component of vector $\hat{\boldsymbol{y}}$.

4.1.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are widely used in image processing tasks. These have been shown to be more robust to overfitting as they have fewer parameters per layer compared with dense layers. CNNs use parameter sharing, local connectivity and pooling [49], which, combined, make them especially useful as feature extractors. Figure 4.4 illustrates a typical architecture of a CNN.

In each convolutional layer, a set of filters are applied to the output of the previous layer, which consists of a 3D tensor (width, height and number of channels). These filters are called *kernels* and basically consist on 3D tensors, typically of shape height $\times$ width $= 3 \times 3$ and depth equal to the output of previous layer (also current layer's input).

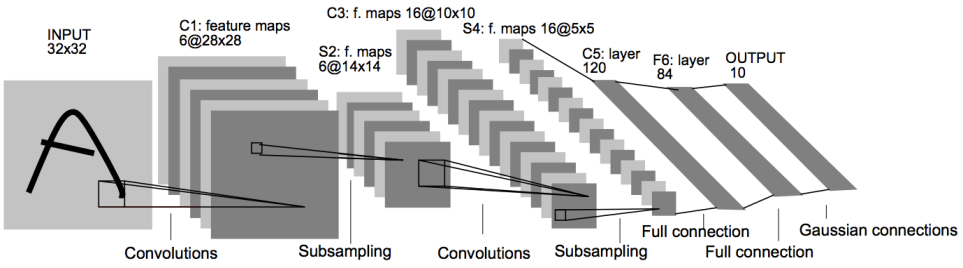


Figure 4.4: Architecture from LeNet-5 [50]. They used it for digit recognition. In it, we observe two stages: (i) convolutional and (ii) dense. The former stage has 4 convolutional layers which try to extract relevant features from the input image. The planes in each layer are known as *feature maps* and are the result of applying a certain filter to the output of the previous layer. The latter (previously explained), uses this "new representation" of the input data to classify the input image by means of linear combinations and non-linear activation functions.

The kernels are slid through the layer's input performing, at each pixel location (i_0, j_0), the following operation

$$\sum_{i=0}^{H}\sum_{j=0}^{W}\sum_{k=0}^{C} W_{i,j,k} X_{i_0-i'+i, j_0-j'+j, k}, \tag{4.9}$$

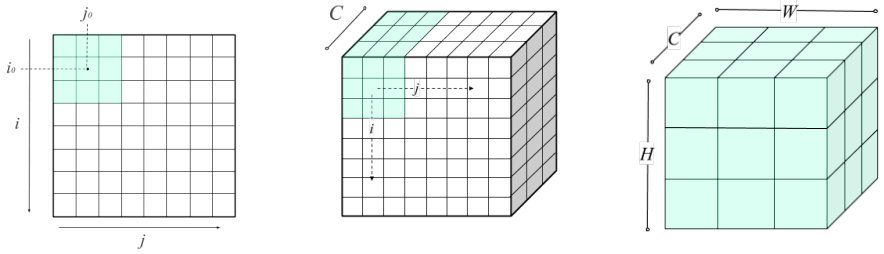
where H, W and C are the height, width and depth of the kernel (depth is also the same in the layer input), i' and j' are such to place the kernel centre-wise, i.e. $i' =$ and $j' =$. Figure 4.5 illustrates a 3×3 kernel sliding through a 3-channel image (e.g. RGB image).

The value obtained from (4.9) is then passed through an activation function, typically the ReLU defined in (4.7).

Pooling layer

Another important component of convolutional networks is the *pooling layers*, which simply reduce the dimensionality of layer outputs. This has to major impacts:

- **Dimension reduction**: The number of network parameters is reduced, which accelerates the convergence of the optimisation algorithm and prevents from overfitting (more on this to come).



(a) Kernel being applied to a single channel.

(b) 3D view. We observe that the kernel always covers the tensor's depth.

(c) $3 \times 3 \times 3$ kernel, often simply 3×3 as the depth is implicit.

Figure 4.5: Each kernel is slid through the layer input tensor along its width and height. As it slides, it performs the operation given by (4.9), to ultimately generate the input for next layer. Each kernel generates a feature map, which combined with the remaining feature maps builds a tensor with a depth equal to the number of kernels.

- **Rotational/Translation invariance**: Given an $H \times W$ grid, it extracts the "most relevant" feature irrespective of its position. Since pooling does not capture the position of such feature, we can affirm that it provides rotational/positional invariant feature extraction.

There are several pooling techniques, but in this work we have used max pooling [51]. The pooling layer is usually placed after the convolutional yet before the activation function, as shown in figure 4.6.

4.2 Training

Machine learning models, in general, are based on the capacity to apply learned knowledge to new situations. Such learned knowledge is obtained from a so-called *training set* and stored as weight matrices in ANNs. Training a machine learning model consists of three major steps:

- **Forward pass:** The model operation (say the one from(4.4)) is applied to a set of samples.

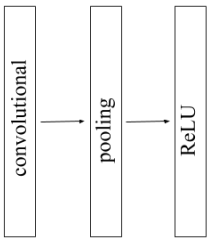


Figure 4.6: A standard approach to build a CNN nowadays is to concatenate blocks such as the one presented in the figure. That is, a convolutional layer which performs a set of summations and multiplications, followed by a pooling layer that reduces the dimensionality of the forward-propagating data tensor and an activation function, often ReLU.

- **Backward pass:** The model estimations are compared with the ground truth values by means of an *objective function* or *loss function*.
- **Update pass:** The model parameters (weight matrices in ANNs) are updated based on the feedback received in the backward pass.

All of these steps are necessary in order to get an ANN to perform some task. The main assumption in this is that unseen data is drawn from the same distribution as the training data. Factors such as the training data size and the complexity of the model (number of parameters, number of operations ...) affect the generalisation of the learned model. Memorising the training data instead of actually learning the underlying function may lead to *overfitting*, which yields to good model performance on training data but very poor performance on unseen data. Techniques to overcome the problem of overfitting are known as *regularisation techniques*, but before that, we need to introduce the idea of *loss function*.

4.2.1 Loss Function

To optimise a model and find good model parameters we define a loss function $\mathcal{L}$ which aims to measure the poorness of the model's performance. As we train our model, we want this parameter to be

minimised, which is accomplished by setting the derivative of the loss function to zero. In general, we define the loss over a batch of N samples $\mathbf{X}^{[N]}$ as

$$\mathcal{L}(\mathbf{X}^{[N]}) = -\frac{1}{N}\sum_{n=0}^{N-1} L(\boldsymbol{t}^{(n)}, \hat{\boldsymbol{t}}^{(n)}), \tag{4.10}$$

where $L(\cdot,\cdot)$ denotes the loss function, which may vary depending on the task, $t^{(n)}$ and $\hat{t}^{(n)}$ are the ground truth value and model prediction for sample n.

Binary cross-entropy loss

For binary classification tasks, such as the one presented in our first experiment, the binary cross-entropy loss is used, defined as

$$L(t,\hat{t}) = -t\ \log\hat{t} + (1-t)\log(1-\hat{t}), \tag{4.11}$$

where t and $\hat{t}$ are the target label (0 or 1) and network prediction for an input sample, respectively, which in this case come as scalar values.

Multiclass cross-entropy loss

For a classifier the multiclass cross-entropy loss is used instead, defined as

$$L(\boldsymbol{t},\hat{\boldsymbol{t}}) = -\log(\hat{t}_{class(\boldsymbol{x})}), \tag{4.12}$$

where $\boldsymbol{t}$ and $\hat{\boldsymbol{t}}$ are the one-hot-encoded ground truth label and model prediction and $\hat{t}_{class(\boldsymbol{x})}$ is the element of $\hat{\boldsymbol{t}}$ corresponding to the class of $\boldsymbol{x}$. Note that, ideally, we want this element to be close to 1 while the rest close to zero (as representend by the one-hot-encoded label $\boldsymbol{t}$).

Mean square error

For regression tasks, such as the one we will present in experiment 3, the Mean Square Error (MSE), shown in (4.13), is a simple, yet effective, objective function.

$$L(t,y) = |t-\hat{t}|^2, \tag{4.13}$$

where here t and $\hat{t}$ are regular vectors, meaning that they are not one-hot-encoded representations.

4.2.2 Regularization

As aforementioned, overfitting is a latent problem in machine learning. Regularization techniques can come in multiple forms. Its core idea is to limit the degrees of freedom of our model to prevent the model to be fitted to very complex patterns, which essentially may appear due to noise in the data, or irrelevant patterns. One way of implementing this concept is by introducing a penalty coefficient in the loss function as in (4.14) which essentially constrains the magnitude of the weight matrices.

$$\mathcal{L}(\mathbf{X}^{[N]}, \mathbf{W}, \lambda) = -\frac{1}{N}\sum_{n=0}^{N-1} L(t_n, y_n) + r(\mathbf{W}), \tag{4.14}$$

where λ is a hyperparameter and $r(\mathbf{W})$ is the summation of all model weights squared. In convolutional networks, for instance, it is defined as $r(\mathbf{W}) = \sum_k \sum_l \sum_i \sum_j (W_{i,j}^{(k,l)})^2$ with $W_{i,j}^{(k,l)}$ denoting the weight (i, j) from kernel l in layer k.

Dropout

Another powerful regularisation technique is dropout [52], introduced by [refernece] and basically consists of randomly setting some activations to zero. A very concise yet simple definition of dropout has been recently given by [53]:

> *[...]. Dropout, simply described, is the concept that if you can learn how to do a task repeatedly whilst drunk, you should be able to do the task even better when sober. [...]*

Batch Normalisation

Batch normalisation [54] aims to make each layer's output Gaussian distributed. To this end, while training, it aims to centre and normalise the data given some estimates on the mean and variance parameters of the, supposed, data distribution. When it comes to the optimisation algorithm, it has been proven to reduce the dependence on the model initial parameters.

4.2.3 Optimiser

Once the loss has been computed for a set of samples, it is time to update the model's parameters accordingly. As aforementioned, minimizing the loss function is a convex problem which implies an optimisation process where different techniques may be used. The standard one is Stochastic Gradient Descent (SGD), which, although introduced many years ago, is still in use nowadays with some minor modifications.

Stochastic Gradient Descent

Given a coordinate point, the gradient of a function points to the direction where the function has its steepest slope at that particular point. SGD takes advantage of this notion and tries to move the towards the opposite direction, such that the objective function is minimised. This is the idea behind most of the optimisers used nowadays. SGD can be summarised by (4.15), where the point $\boldsymbol{x}^{(t)}$ is moved in the opposite direction than the gradient ($-\nabla_x f(\boldsymbol{x}^{(t)})$, where ∇ stands for the gradient operator) at a pace given by the learning rate η giving birth to an update version of the previous point, i.e. $\boldsymbol{x}^{(t+1)}$.

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \nabla_x f(\boldsymbol{x}^{(t)}). \tag{4.15}$$

Momentum

This was proposed as a means to dampen the update vector as the gradient is constantly oscillating. This results in more rapid convergence.

$$\begin{aligned} \boldsymbol{v}^{(t+1)} &= \gamma \cdot \boldsymbol{v}^{(t)} + \eta \nabla_x f(\boldsymbol{x}^{(t)}) \\ \boldsymbol{x}^{(t+1)} &= \boldsymbol{x}^{(t)} - \boldsymbol{v}^{(t+1)}. \end{aligned} \tag{4.16}$$

This optimiser, is the one that has been used in the first experiment.

Adaptive Learning Rates

It is often the case that larger or smaller update steps are needed depending on the "landscape" of the loss function. To this end, several adaptive methods have been proposed such that the updates are

adapted to each individual parameter. A simple approach has been to reduce the learning rate η as we iterate, i.e.

$$\eta_n = \frac{\eta_0}{1 + \alpha n}, \tag{4.17}$$

where η_0 is the initial learning rate, η_n is the learning rate at epoch n and α is the learning decay. More complex adaptive methods are used nowadays, which include RMSProp [55], AdaDelta [56] or ADAM [57] optimisers. This latter is the one used in experiments 2 and 3.

Chapter 5

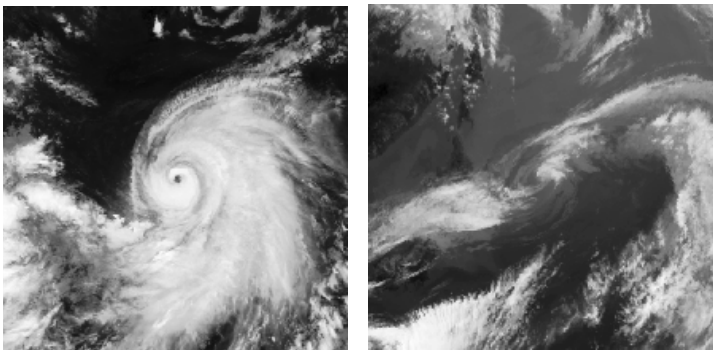
Experiments

In this chapter, we will discuss the main experiments of this work together with the results obtained. First, we describe a CNN-binary classifier for Tropical Cyclones (TCs) and Extra-Tropical Cyclones (ETCs). Next, we explore different CNN architectures to estimate the pressure of a typhoon.

5.1 Tropical Cyclone vs Extratropical Cylone

As previously explained in Table 3.2, Digital Typhoon samples are categorised in 5 different categories. On the one hand, categories 2-5 represent different intensity levels of TCs. On the other hand, category 6 stands for a different natural phenomenon known as ETC. Many TCs transform into ETCs at the end of their "lives", a process which is known as extratropical transition [58]. Thus, prior to further analyse the dataset imagery, we found essential to design an algorithm to effectively distinguish typhoons from extratropical cyclones using satellite images in an automated manner. The fact that both phenomena require different forecast strategies [58] further reinforced the necessity of implementing such classifier. To this end, we implemented a binary classifier using CNNs. In addition, we believe that such algorithm may shed some light on the patterns that best characterise and distinguish these natural phenomena, especially when it comes to the transition from a TC into an ETC. Figure 5.1 shows two example samples from the both classes.

As we observe, the cloud patterns from both phenomena appear to



(a) Tropical cyclone (b) Extratropical cyclone

Figure 5.1: Image data from typhoon sequence 200003, taken in July 2000. As the typhoon evolves over time, it may transition from a TC into a ETC.

be differentiable from the human eye.

5.1.1 Dataset Generation

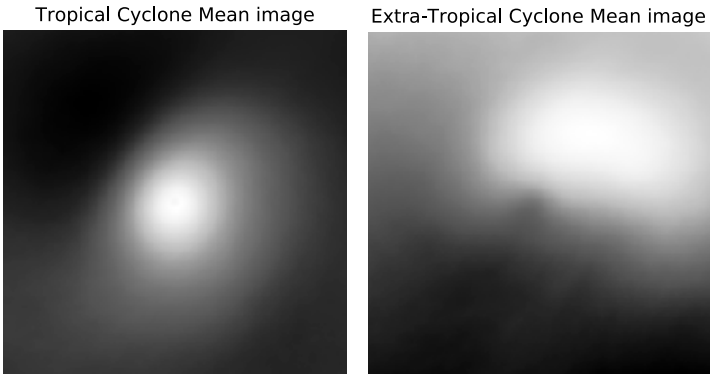
Two main groups (classes) of data are considered:

- **`TC` (0)**: Tropical Cyclone samples, i.e. from categories 2-5.
- **`ETC` (1)**: Extratropical Cyclone samples, i.e. from category 6.

Simply taking all the samples from the Digital Typhoon dataset would lead to a highly unbalanced class distribution, as only around 11% of all the samples belong to class ETC. Therefore, we have built this dataset by considering all ETC samples (approximately 19,000 samples) and an equal amount of TC samples with balanced representation of categories 2-5. All in all, the dataset for this experiment contained around 37,000 infrared labelled images. Using an unbalanced dataset might also provide good results, as the prior information is preserved. However, more research on this is needed.

To get a glance at the complexity of the task, we first obtained the image mean for both classes. These are computed by averaging, pixel-wise, the training images of the respective classes. The results are shown in figure 5.2 and seem to indicate that TC and ETC samples have well distinguished visual patterns. In particular, we observe that

ETCs tend to have a more asymmetrical structure, as a result of the shape distortion suffered while transitioning from a TC to an ETC.



(a) Mean image for TC-labeled training data

(b) Mean image for ETC-labeled training data

Figure 5.2: Looking at the mean images of the TC and ETC-labeled training data we can easily infer that both classes should be easily separable. Dark areas correspond to high-temperature regions and white to low-temperature regions. Figure 5.2a illustrates that the warmest regions are found around the typhoon eye, at the centre of the image. On the contrary, in figure 5.2b we observe the opposite, i.e. lower temperatures are found in the centre. In addition, for this latter class, we observe that highest temperatures are found in the north-eastern side.

5.1.2 Method

As aforementioned, we will be using CNNs to extract features from typhoon imagery. In addition, data processing of the data is required so as to ensure that the model input data is not ill-conditioned.

Data split

60% of the data was used for training, 20% for validation and 20% for testing. We split the data such that data prior to 2011 is used for training and data since that date is used for testing. This is mainly done for two reasons:

- **Real problem conditions**: In real conditions, we usually try to estimate current typhoon characteristics based on previous events.

In addition, when analysing a typhoon image, images from the same sequence are usually not labelled.

- **Similarity between nearby images**: By splitting the dataset this way we ensure that all images belonging to the same typhoon sequence are maintained in the same set. We note that images close in time from the same typhoon sequence are very similar. In fact, they could be considered to be almost equal with small pixel variations. Therefore, the performance obtained using this setting could be misleading, as the model might be overfitting to the training data, which contains samples also appearing (although with small changes) in the validation set.

We will come back to this in section 5.2

Pre-processing

For this experiment, image samples have been centred and normalised using the image mean and the maximum and minimum pixel values. Furthermore, we have used 256×256 resized images, with resolution $1\text{pixel} \approx 10\text{km}$.

$$\frac{X_{i,j} - \bar{X}_{i,j}}{\sigma}, \quad \text{for } 1 < i, j < s, \tag{5.1}$$

where $X_{i,j}$ is the (i, j) pixel value of an image $\boldsymbol{X}$, $s = 256$ is the width and height of images ($\boldsymbol{X}$ has shape $s \times s$), σ denotes the scaling factor obtained from subtracting the maximum and minimum pixel values (i.e. $\sigma = x_{max} - x_{min}$) and $\bar{X}_{i,j}$ is the (i, j) pixel value of the image mean $\bar{\boldsymbol{X}}$, computed as

$$\bar{X}_{i,j} = \frac{1}{N} \sum_{n=0}^{N-1} X_{i,j}^{(n)}, \tag{5.2}$$

where $\boldsymbol{X}^{(n)}$ is the n:th example of the training dataset and N is the number of samples in the training dataset.

Network Architecture

The network architecture is illustrated in figure 5.3 and further detailed in table 5.1. We have used ReLU activation and batch normalisation after all layers (convolutional and dense) except for the output

unit, where we have used sigmoid activation. Max pooling is used in the convolutional layers as a means to decrease the dimensionality of the image and thus reduce the number of network parameters. We have used dropout in the first dense layer to mitigate the overfitting problem. The network output $y = \boldsymbol{f}(\boldsymbol{X})$ is in the range $[0, 1]$ and the estimated class label $\hat{t}$ is obtained based on thresholding y, i.e.

$$\hat{t} = g(y) = \begin{cases} 0 & \text{if } \boldsymbol{f}(\boldsymbol{X}) < 0.5, \\ 1 & \text{if } \boldsymbol{f}(\boldsymbol{X}) > 0.5, \end{cases} \tag{5.3}$$

where $\boldsymbol{f}$ represents all network operations. Overall, the network has 2,891,707 parameters.

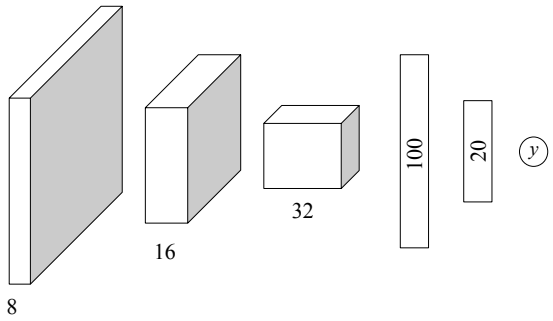


Figure 5.3: The network consists of three convolutional layers, 2 dense layers and one output unit. Details may be found in table 5.1

5.1.3 Training

Due to the binarity nature of the experiment, we used the binary cross-entropy loss function, as defined in 4.11, which was minimised using stochastic gradient descent optimiser with momentum, and mini-batch training with a batch size of 32 samples. According to recent studies batch sizes between 2 and 32 tend to lead to best performance [59]. We tried several hyper-parameter configurations to train the model and evaluated the performance after one full epoch by means of k-fold cross-validation with $k = 8$. This was done in order to select the optimal set of hyper-parameters for training. The best performing configurations are listed in table 5.2.

For the final model, we decided to decrease the learning rate along time, with an initial learning rate $\eta_0 = 0.01$ and rate decay $\alpha = 0.01$

Table 5.1: Building blocks from the Network. We use ReLU activation function and batch normalisation after each layer. Additionally, a max pooling layer is added in all convolutional layers. The first dense layer also has a dropout layer, which intends to mitigate the overfitting problem.

	Convolutional Layers
`L0_conv`	Convolutional, 8 kernels 3×3
`L0_act`	ReLU activation
`L0_bn`	Batch normalisation
`L0_mp`	Max pooling 2×2
`L1_conv`	Convolutional, 16 kernels 3×3
`L1_act`	ReLU activation
`L1_bn`	Batch normalisation
`L1_mp`	Max pooling 2×2
`L2_conv`	Convolutional, 32 kernels 3×3
`L2_act`	ReLU activation
`L2_bn`	Batch normalisation
`L2_mp`	Max pooling 2×2

	Dense Layers
`L3_dense`	Dense, 100 units
`L3_act`	ReLU activation
`L3_bn`	Batch normalisation
`L3_drop`	Dropout 0.2
`L4_dense`	Dense, 50 units
`L4_act`	ReLU activation
`L4_bn`	Batch normalisation
`L5_dense`	Dense, 1 unit
`out`	Sigmoid activation (output)

Table 5.2: k-fold average accuracy of the three best performing hyperparameter configurations using model from figure 5.3 after 2 epochs. We tested momentum values $\mu \in \{0.9, 0.95, 0.99\}$, learning rates $\eta \in \{0.1, 0.01, 0.001\}$ and dropout probabilities $p \in \{0.1, 0.2, 0.5, 1.0\}$. Accuracy is computed after one training epoch.

Learning Rate	**Momentum**	**Dropout**	**Accuracy (%)**
0.01	0.9	0.2	94.5534
0.01	0.9	1.0	94.4389
0.1	0.9	0.2	94.4040

(as in (4.17)). Additionally, we used momentum $\mu = 0.9$ and dropout $p = 0.2$.

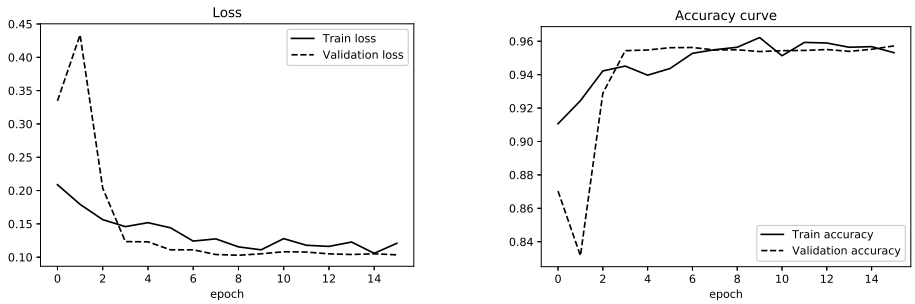
5.1.4 Results

We use early stop based on the validation loss. The best performance model is obtained at epoch 8, with a validation accuracy of 94.73%. This rapid convergence may be explained by the large amount of data at our disposal (approx. 37,000) for a fairly simple task (binary classification). Figure 5.4 illustrates the loss and accuracy curves for the training and validation sets. Test accuracy was also computed, achieving an accuracy of 96.10%. Table 5.3 shows the confusion matrix on the test data. We observe higher values in the matrix diagonal, which quan-

tify the number of correctly classified samples. We obtained precision 97.3% and recall 92.3%, which denotes that there is an asymmetry due to the network misclassifying more TC samples as ETC. One of the reasons for this phenomenon might be the human error when labelling. However, more experiments are to be done to further understand this phenomenon.

Table 5.3: Confusion matrix on test samples. Row i, column j provides the number of samples of class i predicted to be of class j. Class 0 are TC and class 1 are ETC samples.

	predicted TC	**predicted ETC**
TC	2618	202
ETC	89	2716



(a) Loss curve for training and validation sets.

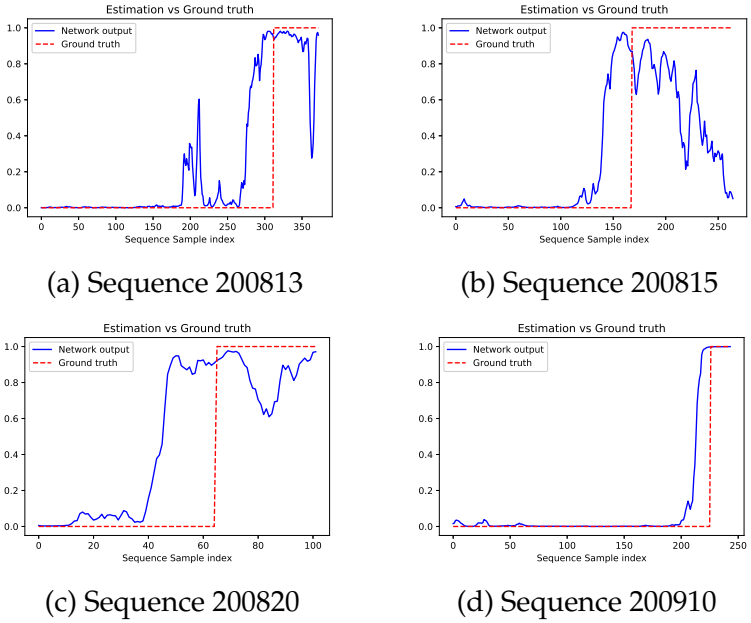
(b) Accuracy curve for training and validation sets.

Figure 5.4: Training and validation accuracy and loss curves.

TC to ETC transition

As aforementioned, a tropical cyclone may, in some cases, transition into an extratropical cyclone. In this sense, we explored the temporal evolution of the obtained predictions to observe how the transition from TC to ETC is done by the network. Figure 5.5 shows some sequence examples. We note that the network tends to estimate that the typhoon is an ETC state before it actually is (according to ground truth data). This is consistent with the results from Table 5.3, which illustrated the model bias towards ETC class. We note that network output tends to oscillate, sometimes producing unexpected results, like

2-hour transitions from ETC to TC and back to ETC. This could be further smoothed by using time series models.



(a) Sequence 200813

(b) Sequence 200815

(c) Sequence 200820

(d) Sequence 200910

Figure 5.5: Network output probability and ground truth class labels. Network tends to anticipate the transition from TC to ETC.

From figure 5.5 we observe that the model tends to predict the transition before the ground truth indicates it. To measure this bias, we looked at the time difference (in hours) between the transition in ground truth and prediction values. This is shown in figure 5.6, which clearly illustrates this bias. In particular, on average, the model predicts the transition 11.14 hours before. We believe that the fact that all ETC samples are preserved and some TC samples are discarded may be a reason for this. However, human mislabelling and other factors may also play a role.

Kernel weights

Figure 5.7 illustrates the 3×3 kernels of the first convolutional layer, i.e. `L0_conv`. In these, we can observe that they have diverse configurations, which make us deduce that each kernel might be looking for different shapes in the input image.

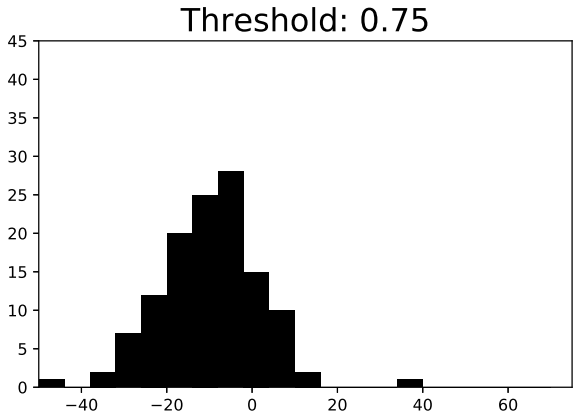


Figure 5.6: To compute the time difference, we assumed that model output greater than 0.75 means that the transition to ETC has been done.

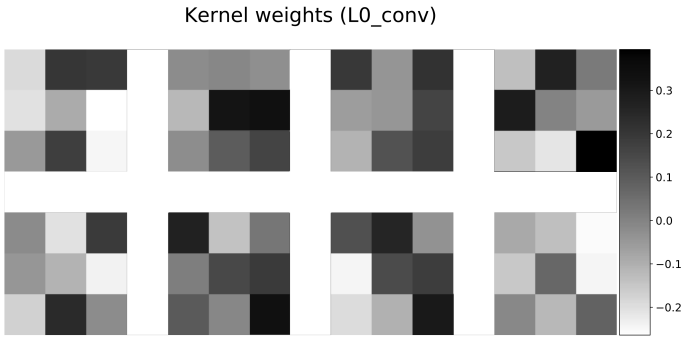


Figure 5.7: The eight kernels of `L0_conv` layer. While kernels in higher levels can also be visualised, their interpretability is more limited.

Network activations

One of the simplest and most straight-forward visualisation technique resides in exploring the activations in the network for different input data. Figures 5.8 and 5.9 illustrate this for a TC and an ETC sample, respectively. We observe that in the first layers some of the activation maps are practically identical. Nonetheless, as we go to higher layers these start to differ. By looking at the activations for both examples we can try to interpret what patterns the kernels are looking for. In layer

`L2_mp` activations (1,4)[9], (3,1) and (4,4) highlight the most characteristic cloud patterns in TCs. Activation (4,2) seems to detect, to some extent, rapid changes in temperatures (cloud borders, typhoon eye) and provides a very decent segmentation of the clouds. In layer `L3_mp` we appreciate that kernels are searching for more detailed cloud structures.

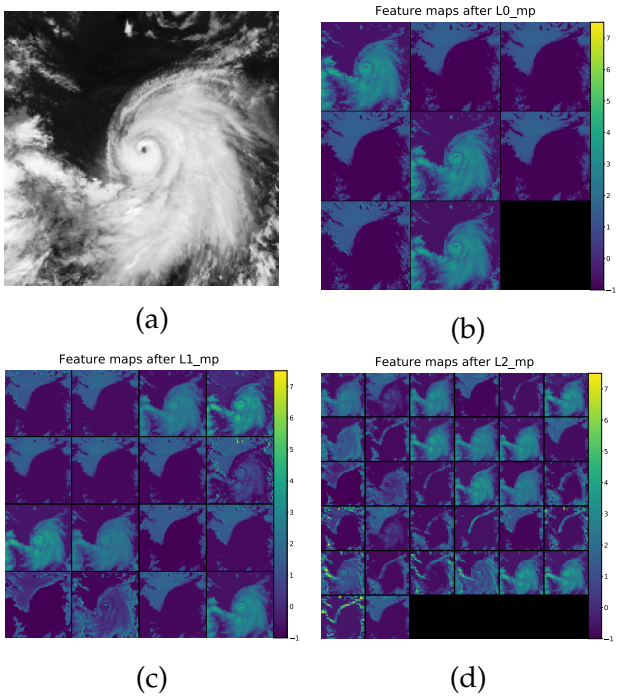


Figure 5.8: Network activations for a TC image sample

Feature attributes

Attribution or *saliency maps* study which regions of an example image are responsible for a specific activation. In our case, we explore how occlusion in the input image affects the output of the network. To this end, we placed a square patch of equal values on different regions of the input image, which occluded its real values. We tested three different values for the patch, (i) minimum value, (ii) mean value and (iii) maximum value of the input example image and visualise the value of the network output depending on the location of the patch. Figure

[9]We use notation (m, n) to refer to the m:th row and n:th column in a grid of images.

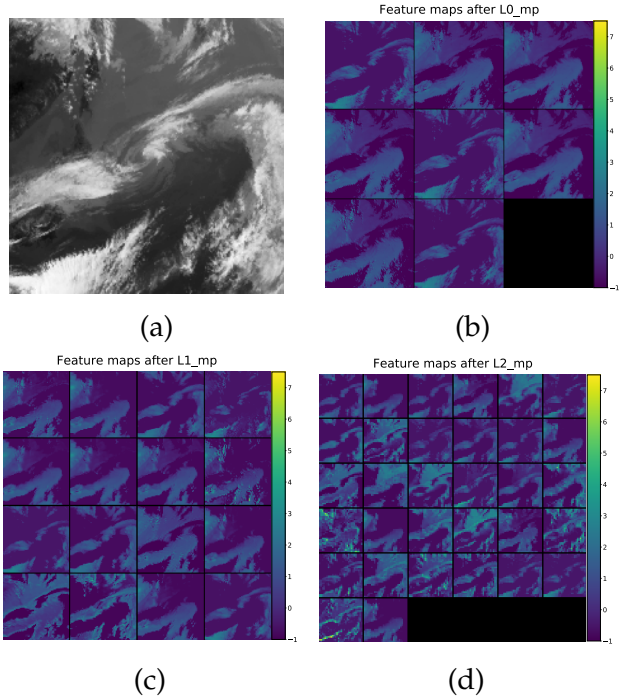


Figure 5.9: Network activations for a ETC image sample

5.10 illustrates the results for TC and ETC examples. Overall, we notice that our model is more robust to the effect of the patch in the case of TC examples while it tends to be more sensible when the input sample is from class ETC. We note that we used a bigger patch for the TC case, otherwise the effect of the patch was neglectable. For TC examples, we observe that the patch most affects the network's prediction when it has values close to the image mean, as shown by figure 5.10b.

In figures 5.10g and 5.10j we observe that the output of the network tends to indicate that the image should be classified as a TC (when it is actually an ETC) when the patch uses negative values and it is placed close the image centre. This might be due to the fact that TCs have low temperatures around the typhoon eye, which is, indeed, located in the image centre. Hence, the network seems to be missing the typhoon eye information, as the patch does not recreate this effect.

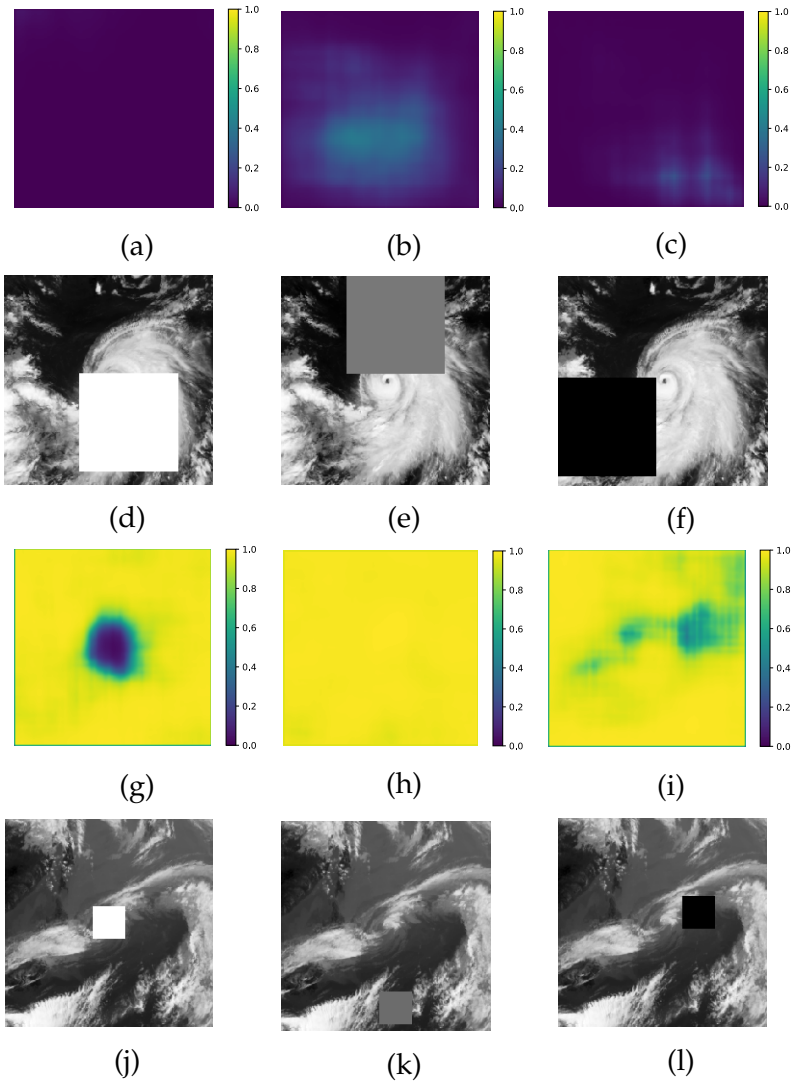


Figure 5.10: First two rows correspond to a TC input example and last two rows correspond to an ETC input example. First and third row: These heat maps illustrate the network output value depending on the location of the patch. The x and y axis represent the location of the patch in the input image space. For TCs, we want the heatmap to be zero while in ETCs we want it to be one. Second and fourth row: Example of the patch being applied to the input image. It shows the case that leads to the worst estimation.

5.2 Typhoon Intensity Classifier

In this section, we implement a classification model to distinguish between classes 2, 3, 4 and 5 of tropical cyclones, as described in table 3.2. This work has been inspired by the results from Pradhan et al. [17], where authors achieved an accuracy of nearly 80% on an 8-category typhoon imagery classification experiment and Chen [10], where an accuracy of 39% was achieved on a 4-category classifier. In particular, we are interested in finding the reasons behind the large divergence in performance between these works. We speculate that the main reason resides in the way the training and test sets are generated, thus, in this work, we will assess the model performance depending on the dataset split.

Figure 5.11 illustrates the mean images for each of the considered categories. Their similarity contrasts with the clear differences between the mean images of TC and ETC. Therefore, we can guess that this task will be more complex.

5.2.1 Method

Our approach follows the same core ideas of the previous experiment. However, we use a network with slightly more parameters as the task is now a bit more complex.

Data split

As explained above, we will run this experiment using different split configurations to confirm the relevance of dataset split. Below we list the two set ups considered for this task.

- **Year sequence split**: Images from the same typhoon sequence are contained in the same set. That is, if an image from typhoon sequence S_i is placed in the training set, all the rest of images from S_i are also placed in the training set. Moreover, we distribute the typhoon sequence in a way such that we use typhoons prior to 2011 for training (and validation) and posterior to 2011 for testing.
- **Random split**: In this case, we just randomise all images and distribute them among the training, validation and test sets. In this case, images from S_i may appear in different sets.

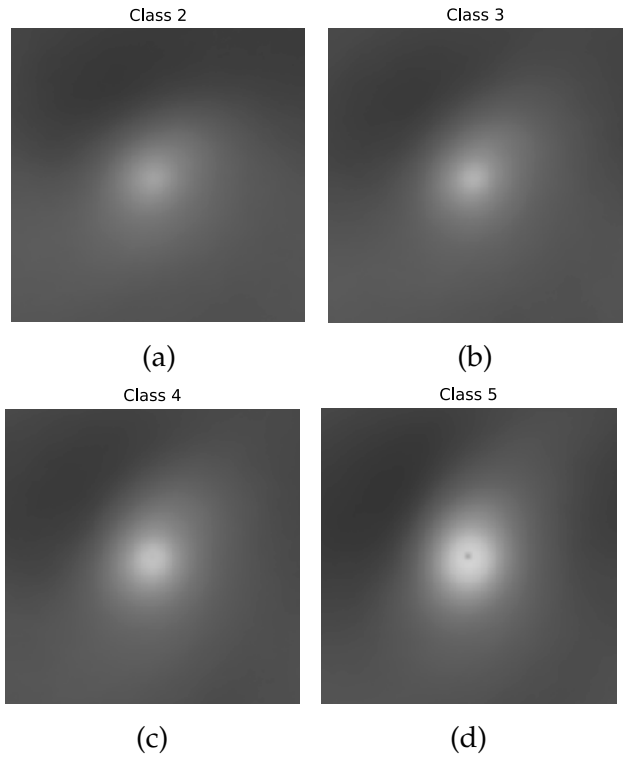


Figure 5.11: Mean images for typhoon intensity categories 2, 3, 4 and 5.

Pre-processing

Pre-processing is done exactly as in the first experiment, i.e. images are centred using image mean and normalised using the difference between the maximum and minimum pixel values in the training set (refer back to 5.1.2 for the details). However, we now crop the images and retain only the centre 128×128 region, as illustrated in 5.12. This reduces the dimensionality of the model input by 75% and eventually removes irrelevant (and thus noisy) image components. This way we reduce the risk of overfitting and ease the convergence of the model.

Network Architecture

Figure 5.13 illustrates the architecture. The overall model architecture is very similar to the one used in the first experiment. The main difference resides in that now more kernels are used in each layer and an extra convolutional layer. In addition, in this experiment, the output

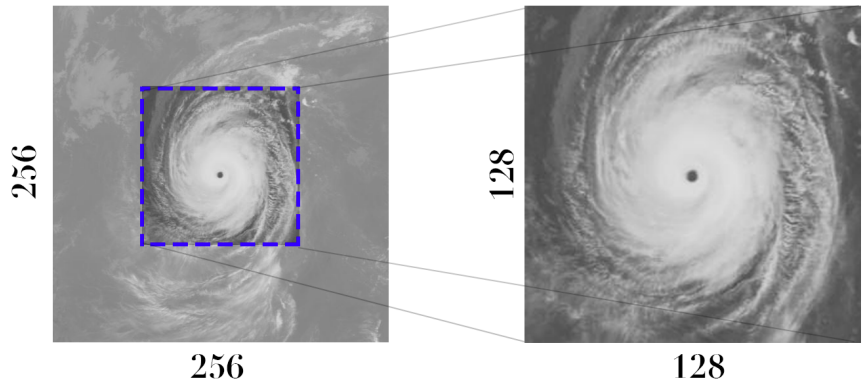


Figure 5.12: The only centre region is used for estimation. We note that all images contain the typhoon eye centred in the image. By cropping we preserve the same resolution as the original image, i.e. 1pixel $\approx$ 10km.

is a four-unit vector to which softmax activation function is applied so as to obtain the probabilities of the input image belonging to each category.

Table 5.4: Building blocks from the Network. We use ReLU activation function and batch normalisation after each layer. Additionally, a max pooling layer is added in all convolutional layers. The first dense layer also has a dropout layer, which intends to mitigate the overfitting problem. Output vector is obtained after applying a softmax layer.

	Convolutional Layers
`L0_conv`	Convolutional, 64 kernels 3×3
`L0_act`	ReLU activation
`L0_bn`	Batch normalisation
`L0_mp`	Max pooling 2×2
`L1_conv`	Convolutional, 128 kernels 3×3
`L1_act`	ReLU activation
`L1_bn`	Batch normalisation
`L1_mp`	Max pooling 2×2
`L2_conv`	Convolutional, 128 kernels 3×3
`L2_act`	ReLU activation
`L2_bn`	Batch normalisation
`L2_mp`	Max pooling 2×2
`L3_conv`	Convolutional, 256 kernels 3×3
`L3_act`	ReLU activation
`L3_bn`	Batch normalisation
`L3_mp`	Max pooling 2×2

	Dense Layers
`L4_dense`	Dense, 1024 units
`L4_act`	ReLU activation
`L4_bn`	Batch normalisation
`L4_drop`	Dropout 0.2
`L5_dense`	Dense, 256 units
`L5_act`	ReLU activation
`L5_bn`	Batch normalisation
`L6_dense`	Dense, 4 units
`out`	Softmax activation (output)

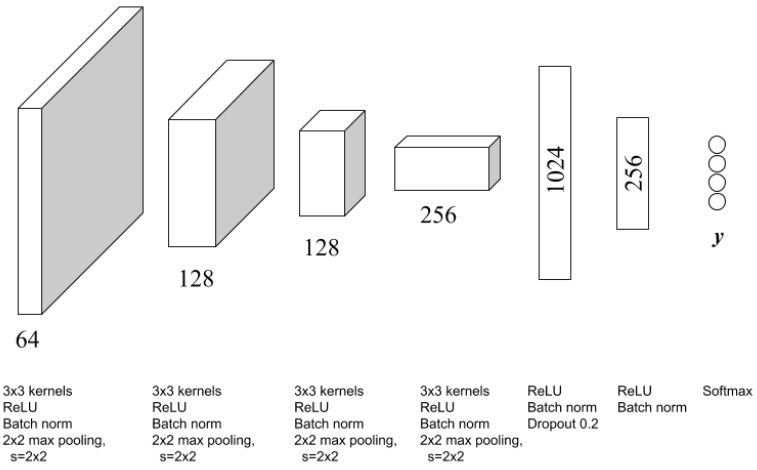


Figure 5.13: The network consists of four convolutional layers, 2 dense layers and four output units. Details may be found in table 5.4.

Several convolutional neural network architectures were tested, and the one provided here seemed to provide the best balance between performance and computational cost. However, we note, again, that the focus on this experiment was to show the impact on the dataset split rather than the performance itself.

5.2.2 Training

We used mini-batch training with 32 samples and Adam optimiser to minimise the cross-entropy loss function, as defined in 4.12.

5.2.3 Results

Using year sequence split, our model was able to achieve an accuracy of 58%. Although far from being a good result, we managed to improve the prior result from [10] which was of 39 % without using time information. The main reason behind this improvement may be the usage of more data and the fact that our model was trained from scratch rather than pre-trained on another dataset. In addition, we note that the model in [10] used nearly 143 million parameters (the pre-trained model was VGG, which is computationally very costly), whereas ours used around 9 million (i.e. nearly a 94% drop). Table 5.5 shows the confusion matrix for validation set data for the two considered dataset

split methodologies. Likewise, table 5.6 shows the precision and recall scores.

Table 5.5: Confusion matrix on validation samples. Row i, column j provides the number of samples of class i predicted to be of class j. On the left, results with year sequence split, on the right, results with a random split.

	Predicted			
	2	**3**	**4**	**5**
2	2876	1012	147	22
3	1082	1736	1037	202
4	239	1087	1865	866
5	37	278	835	2907

	Predicted			
	2	**3**	**4**	**5**
2	5586	769	105	16
3	809	4893	660	114
4	147	1087	5169	514
5	26	278	513	5806

Table 5.6: Precision and recall scores for the validation set data samples. On the left, results with year sequence split, on the right, results with a random split.

	precision	**recall**
2	0.68	0.71
3	0.42	0.43
4	0.48	0.46
5	0.73	0.72
Total	0.58	0.58

	precision	**recall**
2	0.85	0.86
3	0.76	0.76
4	0.80	0.80
5	0.90	0.90
Total	0.83	0.83

As expected, we observe that the performance of our model substantially improves when the datasets are completely randomised. To understand this, we should remember the main underlying assumption in machine learning/deep learning: validation/test data has the same distribution as the training data. This assumption seems to be breaking when year sequence split is used, as the difference between images at sequence level is high. On the contrary, when we randomly split the dataset indistinctly of typhoon sequences all sets (test, validation and training) may contain similar distribution. Figure 5.14 attempts to illustrate this problem.

5.3 Typhoon Centre Pressure Regression

In this last experiment, we attempt to design and implement a deep learning model to provide an estimation of typhoon centre pressure. Centre pressure is an indicator of the intensity of typhoons. Strong

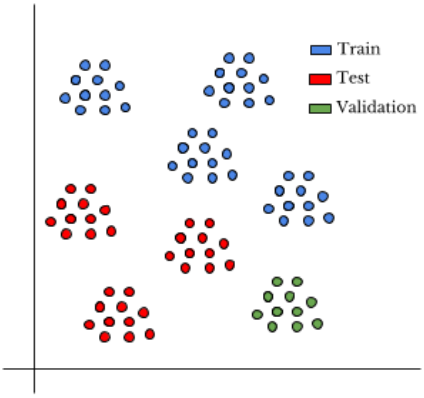


Figure 5.14: Each point in the graph represents an image and each of the clouds would be a typhoon sequence. Training, test and validation may not have similar data distribution when using year sequence split. Making it harder for our model to learn an effective classification scheme. Note that this is a simplification, only meant to illustrate the problem.

typhoons have low centre pressure (e.g. 880 hPa), whereas average typhoon tends to have 980 hPa. Therefore, we find it relevant to devise a model able to predict the centre pressure and thus give us an indicator of the strength of the typhoon under analysis.

5.3.1 Method

To this end, we will use again two stages: (i) A convolutional network to extract relevant features from input images and (ii) densely connected layers to perform the regression itself. The main challenge in this experiment resides in the fact that the distribution of the data is unbalanced (as highlighted in figure 3.3), since most of the samples have pressure values in the range of $[980, 1000]$ hPa. Consequently, we expect our model to have a lower performance as samples move away from this range.

Data split

We use year sequence split, as reasoned in previous experiments.

Pre-processing

Pre-processing is done as in previous experiments, i.e. using the mean image and maximum/minimum pixel values (refer back to 5.1.2 for the details). However, we now crop the images and retain only the centre 128×128 region, as illustrated in 5.12, as in experiment 2.

Network Architecture

The network consists 5 convolutional layers - each of them with ReLU activation, batch normalisation and 2×2 max pooling -, two dense layers - with ReLU activation, dropout with $p = 0.2$ and batch normalisation the first one and with ReLU activation and batch normalisation the last one. Figure 5.15 and table 5.7 provide the details of the architecture.

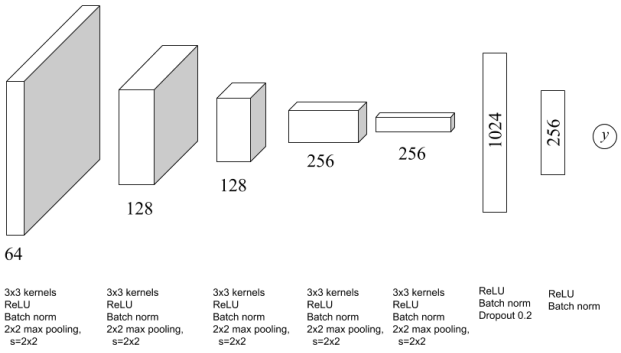


Figure 5.15: The network consists of five convolutional layers, 2 dense layers and four output units. Details may be found in table 5.7.

5.3.2 Training

Our model was trained using Adam optimiser, minimising the mean square error. The model that provided the lowest mean square error on the validation data has been kept.

5.3.3 Results

Since this is a regression task, we no longer can use the accuracy as an indicator of its performance. Instead, we now obtain the Mean Abso-

Table 5.7: Building blocks from the Network. We use ReLU activation function and batch normalisation after each layer. Additionally, a max pooling layer is added in all convolutional layers. The first dense layer also has a dropout layer, which intends to mitigate the overfitting problem. We provide a short

	Convolutional Layers
`L0_conv`	Convolutional, 64 kernels 3×3
`L0_act`	ReLU activation
`L0_bn`	Batch normalisation
`L0_mp`	Max pooling 2×2
`L1_conv`	Convolutional, 128 kernels 3×3
`L1_act`	ReLU activation
`L1_bn`	Batch normalisation
`L1_mp`	Max pooling 2×2
`L2_conv`	Convolutional, 128 kernels 3×3
`L2_act`	ReLU activation
`L2_bn`	Batch normalisation
`L2_mp`	Max pooling 2×2
`L3_conv`	Convolutional, 256 kernels 3×3
`L3_act`	ReLU activation
`L3_bn`	Batch normalisation
`L3_mp`	Max pooling 2×2
`L3_conv`	Convolutional, 256 kernels 3×3
`L3_act`	ReLU activation
`L3_bn`	Batch normalisation
`L3_mp`	Max pooling 2×2

	Dense Layers
`L4_dense`	Dense, 1024 units
`L4_act`	ReLU activation
`L4_bn`	Batch normalisation
`L4_drop`	Dropout 0.2
`L5_dense`	Dense, 256 units
`L5_act`	ReLU activation
`L5_bn`	Batch normalisation
`L6_dense`	Dense, 1 unit

lute Error (MAE) obtained from the predictions as

$$\text{MAE} = \frac{1}{N}\sum_{n=0}^{N-1}|(t-y)|, \tag{5.4}$$

where y and t denote the prediction and ground truth values and N is the number of samples. Our model achieved a MAE of 8.30hPa on validation data and 8.56hPa on test data. To further understand the results and get a wider picture of the model's performance, we visualise the estimations and ground truth values in a scatter plot from figure 5.16. We observe that the error tends to be smaller for pressure values around 980 and 1010 hPa, probably due to a large number of examples within this range. However, we are satisfied about the predictions that the model provides for extreme low-pressure values, as it manages to provide meaningful results despite the lack of examples in these regions.

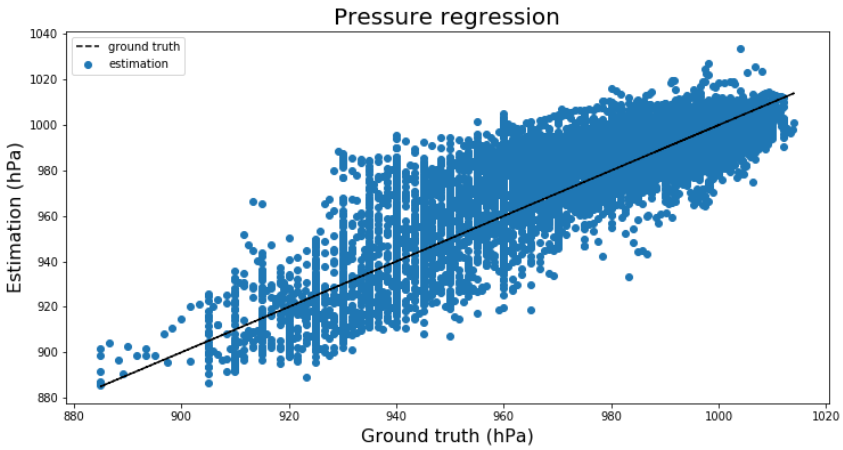


Figure 5.16: The model is able to capture the overall trend. The error tends to increase for lower pressure values, due to data scarcity.

Kernel weights

In this experiment, there are 64 3×3 kernels used in the first layer. From figure 5.17 we observe that most of the kernels seem to be active. However, some others remain inactive. This might indicate that using fewer kernels may lead to similar results.

Network activations

Similar as we did in the first experiment, let us focus on two example images and obtain their corresponding feature map activation throughout the different layers of the network in figure 5.15. To this end, we choose both examples (shown in figure 5.18) to have very different pressure levels.

Figures 5.19 and 5.20 illustrate the feature map activations (high and low pressure, respectively). From these, we can confirm that different maps tend to look for different features. Furthermore, we observe more active maps for the high-pressure example, which might be due to the fact that there the dataset contains more high-pressure samples and, as such, more complex pattern representations are learnt for these samples.

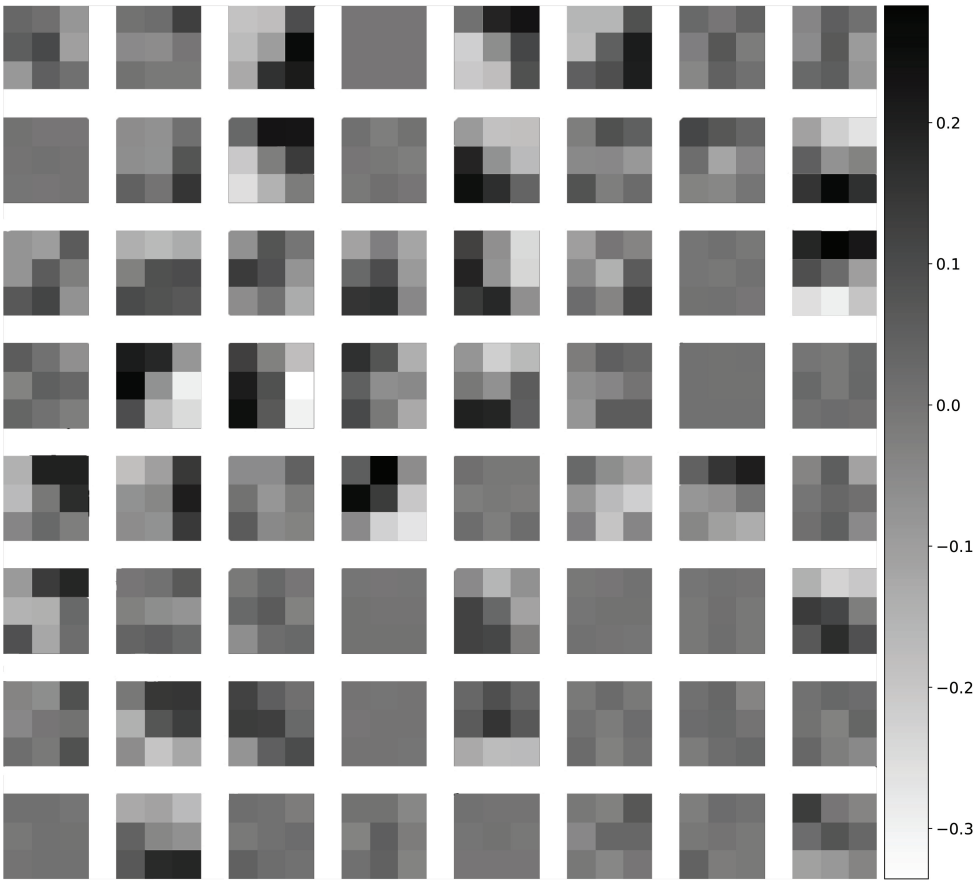


Figure 5.17: Looking at the kernel values we can infer what they might be looking for. For instance, the 8th kernel in the 3rd row and 3rd kernel in 4th-row resemble horizontal and vertical *Sobel–Feldman operator* [60], respectively, which are used to for edge detection.

Use as embedding

Motivated by the results of this experiment, we hypothesise that last layers of the model could be used in other tasks as embeddings of input images. Last layer before output unit contains 256 neurons, as detailed in table 5.7. In particular, we are talking about layer `L5_bn`. Since visualising 256-dimensional data is rather unfeasible, we instead use techniques such as Principal Component Analysis (PCA) [61] or Linear Discriminant Analysis (LDA) [62] to get a glance at the distribution of this last layer.

Figure 5.21 illustrates the cumulative variance ratio retained by

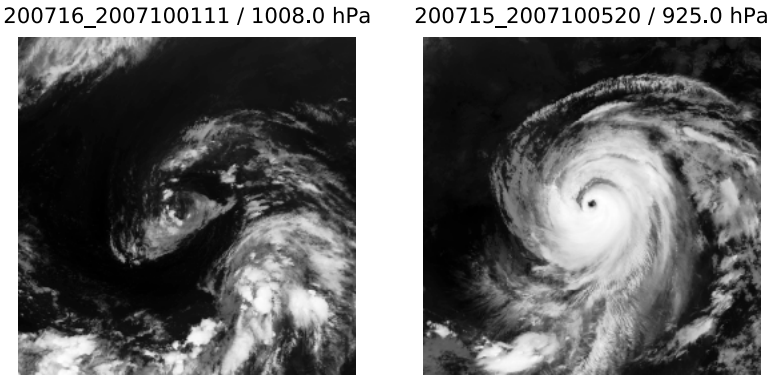


Figure 5.18: On the left an image of a satellite with high centre pressure, on the right one with low centre pressure. Intense typhoon have low centre pressure, as can be observed from these two examples.

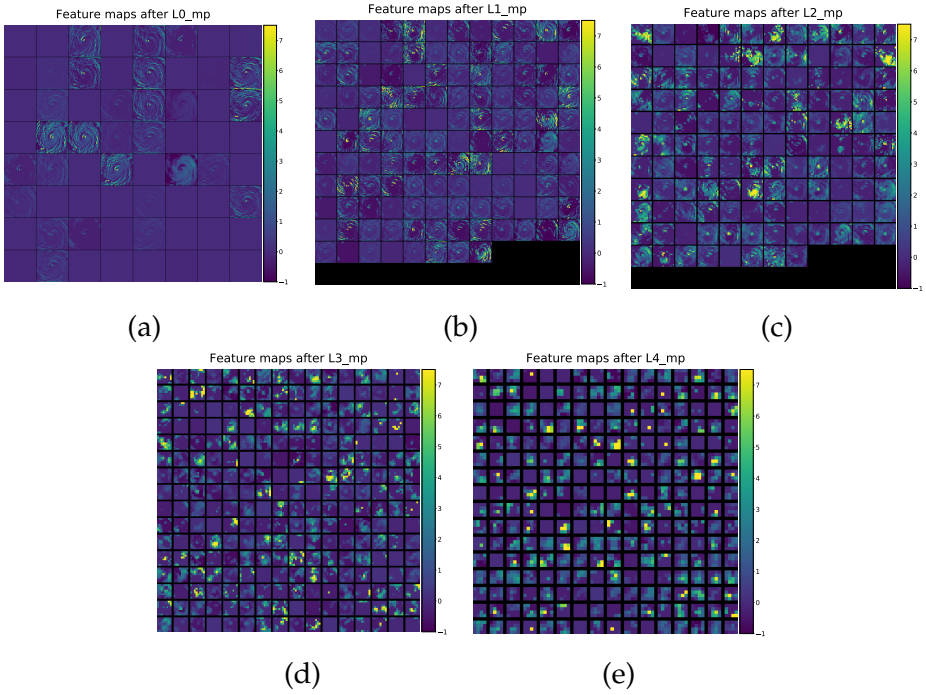


Figure 5.19: Feature maps for the high pressure example from figure 5.18.

each of the PCA components. This can be understood as how much of the information is contained by each of the components. We observe that first three components are responsible for nearly 96% of all information. Hence, we can conclude that the 256-unit representation

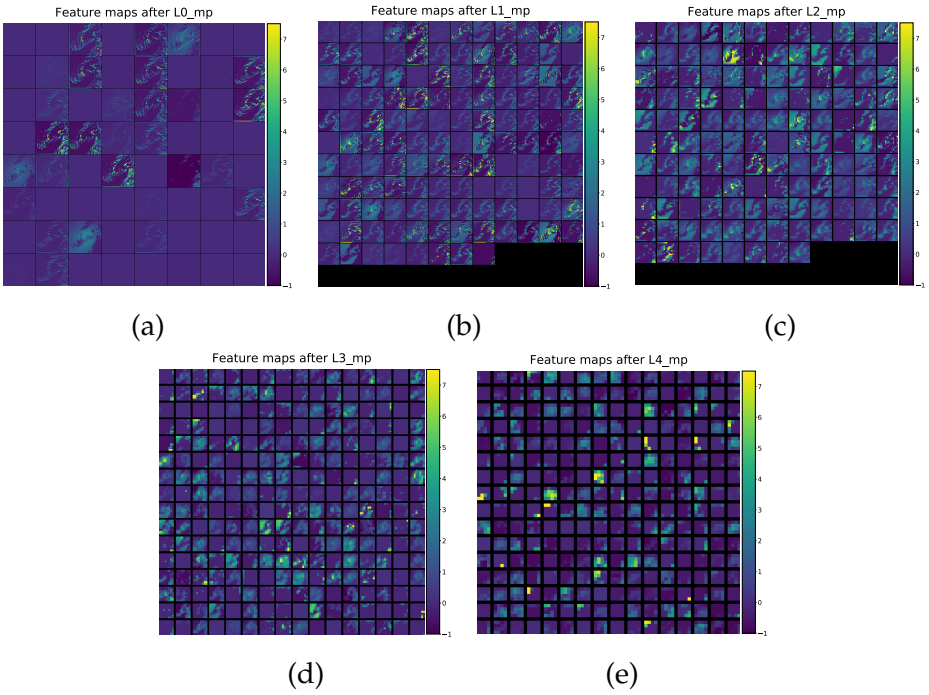


Figure 5.20: Feature maps for the low pressure example from figure 5.18.

in layer `L5_bn` is very sparse.

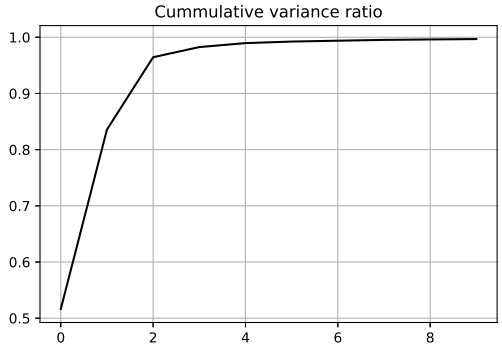


Figure 5.21: Cumulative PCA variance ratio retained by each of component. We note that most of the variance is retained by first three components.

Figure 5.22 shows the PCA component values for each of the samples along the corresponding pressure values. We note that there is a

continuity in the PCA component values as the pressure increases or decreases, which indicates that pressure order is preserved. Whereas 1st PCA component values tend to oscillate more and more as pressure increases, we observe the opposite behaviour for the 2nd PCA component. Hence, we can speculate that both combined could suffice in order to discern different pressure levels. However, we remind the reader that this technique has been used only for visualisation purposes and not to further classify the data, which, however, could be studied.

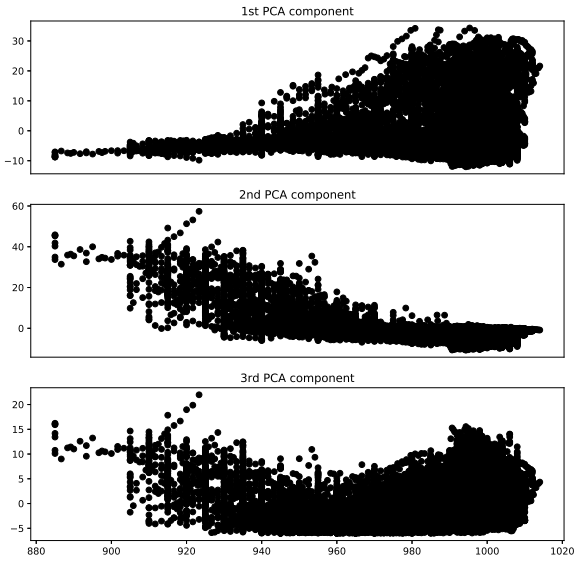


Figure 5.22: x-axis provides the pressure of the test samples and y-axis give their corresponding - first, second or third - PCA component.

Finally, we employ LDA to further obtain more insights through data visualisation. LDA attempts to find a space where inter-class variability is maximised as the intra-class variability is minimised. In other words, samples from the same class should be close to each other whereas samples from different classes should be easily distinguishable. Figure 5.23 illustrates the projection of the data onto the first two LDA components. The colours stand for different pressure intervals, which have been regarded as "classes" by the LDA algorithm. We observe that the data points are somehow clustered and that the intensity

order is preserved, as the first component (x-axis) seems to be proportional to the typhoon intensity (centre pressure).

This opens interesting possibilities, such as new visualisation of typhoon intensity time evolution.

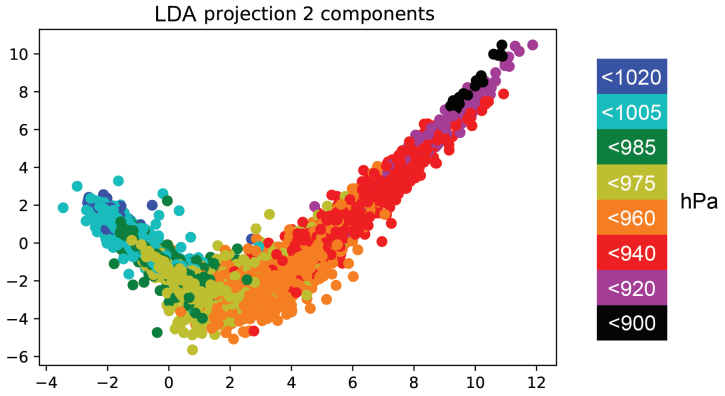


Figure 5.23: First two LDA components of the test samples coloured according to their pressure values.

Chapter 6

Conclusion and Future Work

Throughout this work we have analysed the potential of Deep Learning and Machine Learning applied to the field of Environmental Informatics. Specifically, we have implemented several models able to classify different typhoon categories and predict typhoon centre pressure given their satellite images. Moreover, this project has highlighted an essential and often forgotten aspect of data-driven methods. Namely, when working with real world data, data may be subject to noise and other factors that may corrupt it. Therefore, an exhaustive cleaning should be performed so as to make the data suitable for the experiments. In Chen [10] large amounts of data was dropped because of it being considered as corrupted. In addition, in this work we have worked with images with very specific characteristics. That is, they all preserve certain patterns like, for instance, typhoon eye being always located in the centre of the image. We hypothesise that this preformatting of the images reduces the complexity of the task and cuts down the number of layers that a neural network requires, making very deep networks prone to slow convergence or, even, overfitting.

Overall, we have proven Deep Learning to be an appealing and effective method for the Digital Typhoon project. Nonetheless, we also note that deep learning methods act often as black boxes and may provide unexpected outputs which are hard explain. An example of this could be a typhoon transitionining from ETC to TC back and forth instead of having a smooth transition from TC to ETC, which has been observed to happen with our models.

In this work, the author has attempted to provide clear guidelines on how to work with Digtial Typhoon data and subsequently apply deep learning methods. Nonetheless, there are several experiments that should be explored in the future, such as

- **Unsupervised Learning**: Unsupervised methods are worth exploring, so as to further understand the distribution of the images. Also, this reduced the dependency on human-labeled data, which, in some cases, might add noise to the training.
- **Time information**: We speculate that adding time information of the typhoon sequences to our existing models should eliminate a large number of errors. A standard approach would could be to use RNNs. These, could be used on top of a deep network acting as a feature extractor (e.g. network in experiment *Centre pressure regression* or, alternatively, an unsupervised end-to-end network). However, other approaches are currently being studied in our lab, such as using motion vectors, which also provide powerful temporal information and could be used as an aditional channel in the convolutional network.

6.1 Ethical, Societal and Sustainability aspects

Throughout this work, we have highlighted some of the main motivations for this work. At its core, saving lives is, in the end, the long-term goal of this research. Therefore, we believe that this work has a strong ethical component as it allocates technological resources and scientific knowledge to this end. Kitamoto-sensei started this project (Digital Typhoon) to provide and assist scientists and meteorologists in their attempt to build early warning systems to minimise damage to people, animals, goods cities etc. Natural phenomena will continue to occur whether humans exist or not, thus by preventing such massive losses, we can assure the preservation of us as a society and our ecosystem. Nevertheless, in parallel, we must reduce our impact as humans and society in terms of waste generation, resource abuse to planet Earth.

To sum up, we do need early warning methods, but we also do need love.

Bibliography

[1] Cesare de Federici. "The Voyage and Travaile of M. Caesar Frederick, Merchant of Venice, into the East India, the Indyes, and beyond the Indyes". In: *TransThomas Hickock. London: Richard Jones and Edward White* 1588 (1588).

[2] Thomas R Knutson et al. "Tropical cyclones and climate change". In: *Nature Geoscience* 3.3 (2010), pp. 157–163.

[3] Kerry Emanuel. "Increasing destructiveness of tropical cyclones over the past 30 years". In: *Nature* 436.7051 (2005), pp. 686–688.

[4] Mina Moradi Kordmahalleh et al. "Hurricane Trajectory Prediction via a Sparse Recurrent Neural Network". In: *Proceedings of the 5th International Workshop on Climate Informatics*. 2015.

[5] Seungkyun Hong et al. "Globenet: Convolutional neural networks for typhoon eye tracking from remote sensing imagery". In: *arXiv preprint arXiv:1708.03417* (2017).

[6] Thomas Loridan, Ryan P Crompton, and Eugene Dubossarsky. "A Machine Learning Approach to Modeling Tropical Cyclone Wind Field Uncertainty". In: *Monthly Weather Review* 145.8 (2017), pp. 3203–3221.

[7] SHI Xingjian et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in neural information processing systems*. 2015, pp. 802–810.

[8] John A Knaff et al. "Statistical, 5-day tropical cyclone intensity forecasts derived from climatology and persistence". In: *Weather and Forecasting* 18.1 (2003), pp. 80–92.

[9] John A Knaff, Charles R Sampson, and Mark DeMaria. "An operational statistical typhoon intensity prediction scheme for the western North Pacific". In: *Weather and Forecasting* 20.4 (2005), pp. 688–699.

[10] Danlan Chen. "Typhoon Intensity Forecasts By Using Deep Learning Technique". MA thesis. Montreal, Quebec: School of Computer Science, McGill University, 2017.

[11] *Digital Typhoon oficial website*. digital-typhoon.org. Accessed: 12th May 2018.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[13] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] Yun Li et al. "Leveraging LSTM for rapid intensifications prediction of tropical cyclones." In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 4 (2017).

[15] Ratneel Vikash Deo, Rohitash Chandra, and Anuraganand Sharma. "Stacked transfer learning for tropical cyclone intensity prediction". In: *arXiv preprint arXiv:1708.06539* (2017).

[16] JJ Miller, Manil Maskey, and Todd Berendes. "Using Deep Learning for Tropical Cyclone Intensity Estimation". In: (2017).

[17] Ritesh Pradhan et al. "Tropical Cyclone Intensity Estimation Using a Deep Convolutional Neural Network". In: *IEEE Transactions on Image Processing* 27.2 (2018), pp. 692–702.

[18] Asanobu Kitamoto. "The development of typhoon image database with content-based search". In: *Proceedings of the 1st international symposium on advanced informatics*. 2000, pp. 163–170.

[19] Asanobu Kitamoto. "Enhancing the Quality of Typhoon Image Database Using NOAA AVHRR Data Received in Thailand". In: *Proceedings of the 7th International Workshop on Academic Information Networks and Systems*. 2000, pp. 1–11.

[20] Asanobu Kitamoto and Kinji Ono. "The Collection of Typhoon Image Data and the Establishment of Typhoon Information Databases Under International Research Collaboration between Japan and Thailand". In: (2001).

[21] Asanobu Kitamoto. "Digital typhoon: Near real-time aggregation, recombination and delivery of typhoon-related information". In: *Proceeding of the 4th International Symposium on Digital Earth*. 2005.

[22] A Kitamoto. *Digital Typhoon: Typhoon Analysis based on Artificial intelligence Approach*. Tech. rep. Technical Report of Information Processing Society of Japan, 2000.

[23] A Kitamoto and K Ono. "The Construction of Typhoon Image Collection and its Application to Typhoon Analysis". In: *NII Journal* 1.1 (2000), pp. 7–22.

[24] Asanobu Kitamoto. "Typhoon analysis and data mining with kernel methods". In: *Pattern Recognition with Support Vector Machines* (2002), pp. 511–520.

[25] RK Smith. "Tropical cyclone eye dynamics". In: *Journal of the Atmospheric Sciences* 37.6 (1980), pp. 1227–1232.

[26] Lloyd J Shapiro and Huch E Willoughby. "The response of balanced hurricanes to local sources of heat and momentum". In: *Journal of the Atmospheric Sciences* 39.2 (1982), pp. 378–394.

[27] Vernon F Dvorak. *A technique for the analysis and forecasting of tropical cyclone intensities from satellite pictures*. US Department of Commerce, National Oceanic and Atmospheric Administration, National Environmental Satellite Service, 1972.

[28] Vernon F Dvorak. "Tropical cyclone intensity analysis and forecasting from satellite imagery". In: *Monthly Weather Review* 103.5 (1975), pp. 420–430.

[29] Vernon F Dvorak. *Tropical cyclone intensity analysis using satellite data*. Vol. 11. US Department of Commerce, National Oceanic, Atmospheric Administration, National Environmental Satellite, Data, and Information Service, 1984.

[30] RAYMONDM ZEHR. "Improving objective satellite estimates of tropical cyclone intensity". In: *Conference on Satellite Meteorology and Oceanography, 4 th, San Diego, CA*. 1989.

[31] CS Velden, TL Olander, and MZ Raymond. "Objective Dvorak Technique (ODT)". In: *Weather and Forecasting, Regional and Mesoscale Meteorology Branch, NOAA/NESDIS, US Department of Commerce, Washington, D. C* (1998).

[32] Timothy L Olander, Christopher S Velden, and Michael A Turk. "Development of the Advanced Objective Dvorak Technique (AODT)–Current progress and future directions". In: *25th Conference on Hurricanes and Tropical Meteorology*. 2002, pp. 585–586.

[33] Timothy L Olander and Christopher S Velden. "The advanced Dvorak technique: Continued development of an objective scheme to estimate tropical cyclone intensity using geostationary infrared satellite imagery". In: *Weather and forecasting* 22.2 (2007), pp. 287–298.

[34] Elizabeth A Ritchie et al. "Tropical cyclone intensity estimation in the North Atlantic Basin using an improved deviation angle variance technique". In: vol. 27. 5. 2012, pp. 1264–1277.

[35] Elizabeth A Ritchie et al. "Satellite-derived tropical cyclone intensity in the north pacific ocean using the deviation-angle variance technique". In: vol. 29. 3. 2014, pp. 505–516.

[36] Jan-Hwa Chu. *A Regression Model for the Western North Pacific Tropical Cyclone Intensity Forecast*. Tech. rep. NAVAL RESEARCH LAB MONTEREY CA, 1994.

[37] Kerry Emanuel et al. "Environmental control of tropical cyclone intensity". In: *Journal of the atmospheric sciences* 61.7 (2004), pp. 843–858.

[38] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: 2014.

[39] Matt Menne. "Global long-term mean land and sea surface temperatures". In: 2000.

[40] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: vol. 18. 7. MIT Press, 2006, pp. 1527–1554.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[42] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[43] Robert J Schalkoff. *Artificial neural networks*. Vol. 1. McGraw-Hill New York, 1997.

[44] Stanford University. *Convolutional Neural Networks for Visual Recognition*. URL: http://cs231n.github.io/convolutional-networks/http://cs231n.github.io/convolutional-networks/ (visited on 05/22/2018).

[45] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[46] David Harris and Sarah Harris. "Digital design and computer architecture". In: (2010).

[47] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: (2013), pp. 1310–1318.

[48] John S Bridle. "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition". In: (1990), pp. 227–236.

[49] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[50] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[51] Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: (2010), pp. 92–101.

[52] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[53] Stephen Merity. "The NUGGET Non-Linear Piecewise Activation". In: *arXiv preprint arXiv:1804.404* (2018).

[54] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[55] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[56] Matthew D Zeiler. "ADADELTA: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[57] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[58] Sarah C Jones et al. "The extratropical transition of tropical cyclones: Forecast challenges, current understanding, and future directions". In: *Weather and Forecasting* 18.6 (2003), pp. 1052–1092.

[59] Dominic Masters and Carlo Luschi. "Revisiting Small Batch Training for Deep Neural Networks". In: *arXiv preprint arXiv:1804.07612* (2018).

[60] Irvin Sobel. "An isotropic 3× 3 image gradient operator". In: *Machine vision for three-dimensional scenes* (1990), pp. 376–379.

[61] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[62] Sebastian Mika et al. "Fisher discriminant analysis with kernels". In: *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.* Ieee. 1999, pp. 41–48.